[redacted]

510.84
Il6r
no 287
Cop 2

Report No. 287

DESIGN OF DIGITAL COMPUTER CIRCUITS USING A BASIC LOGIC CELL

by

Harvey Allen Finkelstein

May 24, 1968

DEPARTMENT OF COMPUTER SCIENCE · UNIVERSITY OF ILLINOIS · URBANA, ILLINOIS

Report No. 287

DESIGN OF DIGITAL COMPUTER CIRCUITS USING A BASIC LOGIC CELL*

by

Harvey Allen Finkelstein

May 24, 1968

Department of Computer Science
University of Illinois
Urbana, Illinois 61801

*Submitted in partial fulfillment of the requirements for the Degree of Master of Science in Electrical Engineering.

ACKNOWLEDGEMENT

The author wishes to express his gratitude and to thank his advisor Professor Sylvian R. Ray, Myrna Selivonchick for typing the manuscript, John Otten for preparing the diagrams, and the Digital Computer Laboratory for the reproduction of this thesis, all of whose assistance in the preparation of this paper has been greatly appreciated.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

At present, many digital circuits are built using modules containing two or three NAND gates, NOR gates, or a single flip-flop. The result is that a great many of these modules are usually required to obtain a desired logical function. If these various gates could be combined into a single module or cell capable of performing a large number of operations, dimensional requirements of circuits incorporating these cells would be reduced. Size limitation is extremely important, for example, in digital circuits required in rockets, missiles, and satellites. Therefore, if this multipurpose cell could be designed and constructed at a reasonable cost, a new method of logic design could result.

To achieve the objective of a universal logic cell, a method termed cutpoint logic had been devised. A cutpoint cellular array is a two-dimensional rectangular arrangement of square cells, each of which has binary inputs on the top and left edges and outputs on the bottom and right edges. Each cell is interconnected with neighboring cells, and it is specialized by a set of binary constants that are termed cutpoints.* Although this cellular arrangement represents a step forward in obtaining a reduction in the size of digital circuits, it will be shown in this paper to have several disadvantages. Therefore, two new types of universal cells, the square cell and the hexagonal cell, have been designed to eliminate the failings of the cutpoint cell.

---

*Minnick, R.C., "Cutpoint Cellular Logic", IEEE Transactions on Electronic Computers, December 1964.

The square cell has one fixed input, one fixed output, and two lines that could be programmed either way. The hexagonal cell has three fixed inputs and three fixed outputs. While the cut-point cell can output only functions of two variables, the other two cells yield many of the functions of three variables.

In this thesis, it is suggested that the various cells be programmed by placing an address register in each cell. Microwelding or separate lines could be employed but it will take more extensive examples to justify their use. The addressing will be most efficient if done by a computer. At this time, however, an algorithm to program the hexagonal or square cell has not been found and programming must be done by hand.

The individual cells consist of integrated chips containing NAND and NOR gates and inverters. The number of gates placed on a chip has been limited to 200 which is about the range of our present technology.

Finally, the proposed saving in space and cost requirements are illustrated by various examples of cellular networks. A summation of the number of cells used in applying each of the three methods to specific examples shows that 50% more cells are needed in the cutpoint cases.

Thus, this thesis will attempt to open up a new area in the design of digital computer circuits. By expanding on these methods, cells of different shapes or designs using layers of cells could prove to be even more practical. The final result of the work in this area should be most interesting.

2. CUTPOINT CELL

2.1 Cell Design

The cutpoint cell is a square shaped cell employing two inputs and two outputs. The right output line is tied directly to the left input line. The bottom output yields one of eight functions of two variables or a flip-flop output function.

2.2 Coding

The cell coding and functions performed, determined by a four bit code, are shown in Table 1. No method of addressing each cell other than with 4 switches is given in Mr. Minnick's article. A scheme, however, consisting of a 4 bit register plus the appropriate addressing logic could be used. This is the same method suggested for the hexagonal and square cells and can be found by referring to Sections 3.1 and 3.3.

2.3 Logic Implementation

Two different designs for a cutpoint cell, a resistor-transistor realization and a diode-transistor realization, are given in Mr. Minnick's paper. Hardware design for the cutpoint cell will not be discussed since the cutpoint cell will be shown to be inferior, in a sense to be discussed in the following pages, to the other cells.

2.4 Advantages and Disadvantages

The obvious advantage of designs utilizing cutpoint cells, as well as the hexagonal and square cells to be discussed in the remaining sections, is that many functions are formed in one module. In the cutpoint cell, 8 functions of two variables and a flip-flop function can be performed without using a large number of individual

TABLE 1: CUTPOINT CELL CODING*

0	1
1	$\bar{y}$
2	$\bar{x} + \bar{y}$
3	$\bar{x}\bar{y}$
4	$x + y$
5	$x\bar{y}$
6	$x \oplus y$
7	0
13	$x=S$, $y=R$

*Minnick, R.C., loc. cit., p. 688.

NAND and NOR gates and inverters. The advantage the cutpoint cell has over the square and hexagon is that an algorithm exists to program the required function. This is done by forming each component of the function in an individual column and then gathering up the parts in the last row.

The main disadvantage of the cutpoint cell stems from the large number of cells needed in the programming scheme. Most arrays designed to yield a specific function have as many columns as terms in the function and have one more row than the number of input variables used. Thus, a function

$$f = x_1 x_3 x_4 + x_1 x_2 x_5 + \bar{x}_1 x_5 + x_2 x_6 \qquad (2.1)$$

would be made up in a 4x7 array. Mr. Minnick states that "the synthesis of an arbitrary n-variable combinational switching function is shown to require a cutpoint array n+1 cells high and no more than 2^{n-2} cells wide."* Simplifying the function usually does not work in the cutpoint case since input variables are usually restricted to a single individual row and cannot be added in a subsequent row. Another shortcoming of this cellular method is that it needs two cells just to form all of the functions of two variables whereas all of the functions of three variables are performed by two hexagonal or square cells.

*Minnick, R.C., loc. cit.

3. SQUARE CELL

3.1 Cell Design

A block diagram of a square cell is illustrated in Figure 1. The upper righthand corner shows the address elements, a register control and a 16 bit shift register that sets the cell to the function desired. The number next to the individual blocks indicate which bits control that block and the path that the inputs follow through the cell are shown by the directed lines. Further explanation of this block diagram follows in the next section on coding.

3.2 Coding

The coding of the square cell is shown in Table 2.

The first decision is whether the input T_i is complemented or not and setting q_1 accordingly. Control bits q_2 and q_6 determine whether the sides are inputs and/or outputs and if the sides are not both outputs, the input(s) is either complemented or left alone. Once these preliminaries are completed, the desired output functions are formed. If L is an output, the desired result is obtained by setting q_7, q_8 and q_9 to the proper values. There are 7 possible output functions on this line of which one combination q_7: 1, q_8: 1, and q_9: 0 is set aside to provide more combinations of three inputs on output line B_o. There are two sets of output functions available on the B_o line. The B functions are similiar to the functions provided on the R_o and L_o lines while the A functions are different functions of three variables. The functions

$$(L+\overline{L})\ (T+\overline{T})\ (R+\overline{R})\ +\ (\overline{L}+L)\ (\overline{T}+T)\ (\overline{R}+R) \qquad (3.1)$$

$$L(T\oplus R)\ +\ \overline{L}(T+\overline{T})\ (R+\overline{R}) \qquad (3.2)$$

and

$$L(\overline{T\oplus R})\ +\ \overline{L}(T+\overline{T})\ (R+\overline{R}) \qquad (3.3)$$

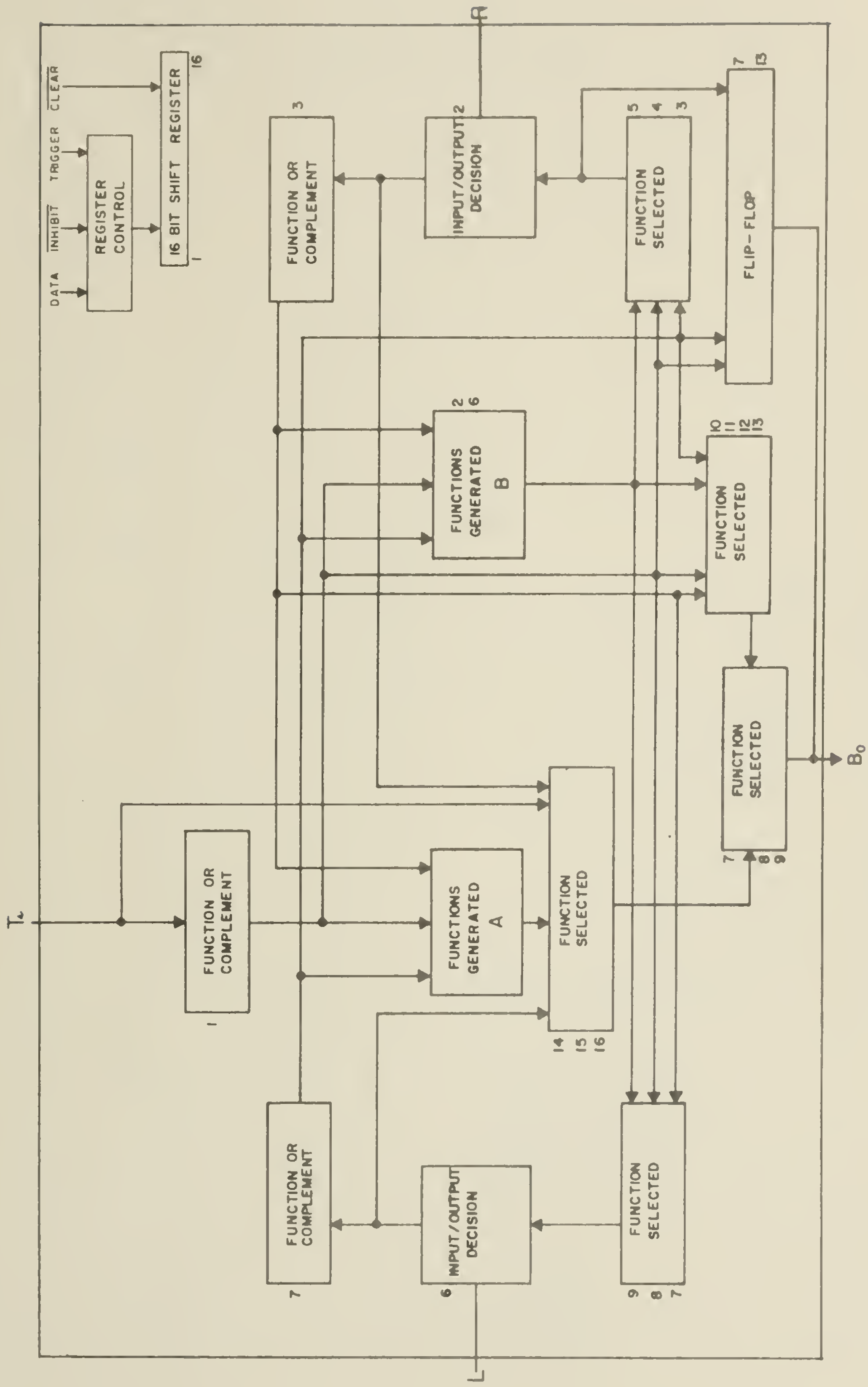


FIGURE I. SQUARE CELL BLOCK DIAGRAM

TABLE 2: SQUARE CELL CONTROLS

q_1:	1	Ti
	0	$\overline{\text{Ti}}$
q_2:	1	Ri
	0	Ro
q_6:	1	Li
	0	Lo

q_6:	1	q_7:	0	$\overline{\text{Li}}$
			1	Li
q_2:	1	q_3:	0	$\overline{\text{Ri}}$
			1	Ri

								FUNCTIONS GENERATED B
q_6:	0	q_7:	0	q_8:	0	q_9:	0	Lo: 1
			0		0		1	$(T+\overline{T})\oplus(R+\overline{R})$
			0		1		0	$(T+\overline{T})+(R+\overline{R})$
			0		1		1	$(T+\overline{T})(R+\overline{R})$
			1		0		0	0
			1		0		1	$R+\overline{R}$
			1		1		0	USE q_{14} q_{15} q_{16}
			1		1		1	$T+\overline{T}$

						FUNCTIONS GENERATED B
q_{14}:	1	q_{15}:	1	q_{16}:	1	
q_{11}:	0	q_{12}:	0	q_{10}:	0	Bo: 1
	0		1		0	$(T+\overline{T})\oplus(L+\overline{L})\oplus(R+\overline{R})$
	1		0		0	$(T+\overline{T})+(L+\overline{L})+(R+\overline{R})$
	1		1		0	$(T+\overline{T})(L+\overline{L})(R+\overline{R})$
	0		0		1	0
	0		1		1	$R+\overline{R}$
	1		0		1	$\text{Li}+\overline{\text{Li}}$
	1		1		1	$T+\overline{T}$

TABLE 2: cont'd

								FUNCTIONS GENERATED B	
q_2:	0	q_3:	0	q_4:	0	q_5:	0	Ro:	1
			0		0		1		$(T+\overline{T})\oplus(L+\overline{L})$
			0		1		0		$(T+\overline{T})+(L+\overline{L})$
			0		1		1		$(T+\overline{T})(L+\overline{L})$
			1		0		0		0
			1		0		1		$(T+\overline{T})\oplus(L+\overline{L})$
			1		1		0		$L+\overline{L}$
			1		1		1		$T+\overline{T}$
q_{13}:	1	$L+\overline{L}$:	1	$T+\overline{T}$:	1	Bo:	0	Ro:	1
			0		1		1		0
			1		0	OUTPUT IS THE SAME			
			0		0	AS PREVIOUS OUTPUT			
q_7:	1	q_8:	1	q_9:	0				
q_{10}:	0	q_{11}:	0	q_{12}:	0	FUNCTIONS GENERATED A			
q_{14}:	0	q_{15}:	0	q_{16}:	0	Bo:	$(\underline{L}+\overline{L})(\underline{T}+\overline{T})(\underline{R}+\overline{R})$ $+(\overline{L}+L)(\overline{T}+T)(\overline{R}+R)$		
	0		0		1		$(R+\overline{R})(T+\overline{T})+(L+\overline{L})$		
	0		1		0		$[(L+\overline{L})+(T+\overline{T})]\ (R+\overline{R})$		
	0		1		1		$[(R+\overline{R})+(T+\overline{T})]\ (L+\overline{L})$		
	1		0		0		$L(T\ominus R)+\overline{L}(T+\overline{T})(R+\overline{R})$		
	1		0		1		$L(\overline{T\oplus R})+\overline{L}(T+\overline{T})(R+\overline{R})$		
	1		1		0		$[(R+\overline{R})\oplus(T+\overline{T})]\ (L+\overline{L})$		
	1		1		1		1		

are included since three cells would otherwise be needed to form these functions. Each of the 256 functions of three variables can be placed in one of 22 different categories of similar functions as listed in Appendix B. Therefore, the remaining function was chosen from the classes of functions of three variables with the most members. If R is an output, the R_o functions are formed using control bits q_3, q_4 and q_5. Finally, if the cell is to be used as a flip-flop, q_{13} is set as a "1", and the left input is the flip-flop set, the top input is the trigger and B_o and R_o are the outputs.

3.3 Logic Implementation

Square cells are designed with the NAND and NOR gates and inverters of Appendix A built onto a single chip or cell. Figure 2 illustrates the decision logic used to set inputs and outputs while Figure 3 shows the logic used to complement the input variables. The three basic functions, the "AND", the "OR", and the "EXCLUSIVE-OR", are formed in Figure 4. Figure 5 contains the right output function selection logic while Figure 6 is for the left output. The special functions are shown in Figure 7 and their selection logic is pictured in Figure 8. Since there is also a flip-flop in each cell, Figure 9 was included.

Finally, there is a need to program the individual cell. Since these cells must be identical, this addressing is accomplished by means of a 16 bit shift register shown in Figure 10. Three of the 4 lines attached to each cell are common to every other cell. The fourth, $\overline{\text{Inhibit}}$, is used to differentiate between cells. A computer is assumed as the source of the 16 bit groups that control each cell's specific function. As each cell's turn to be programmed occurs, the computer would lift the level on the $\overline{\text{Inhibit}}$ line from 0v. to +4v.

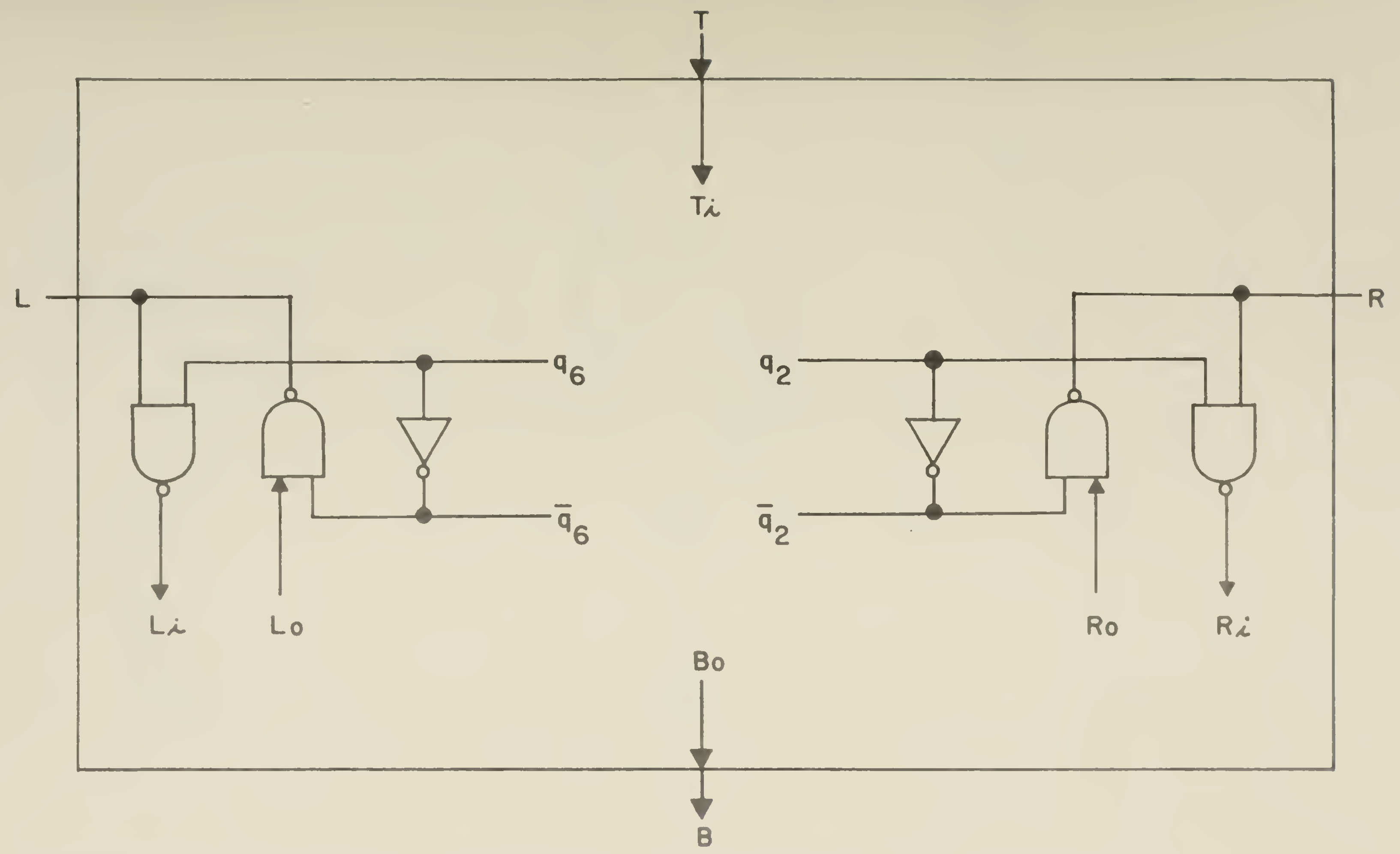


FIGURE 2. SQUARE CELL INPUT-OUTPUT DECISION LOGIC.

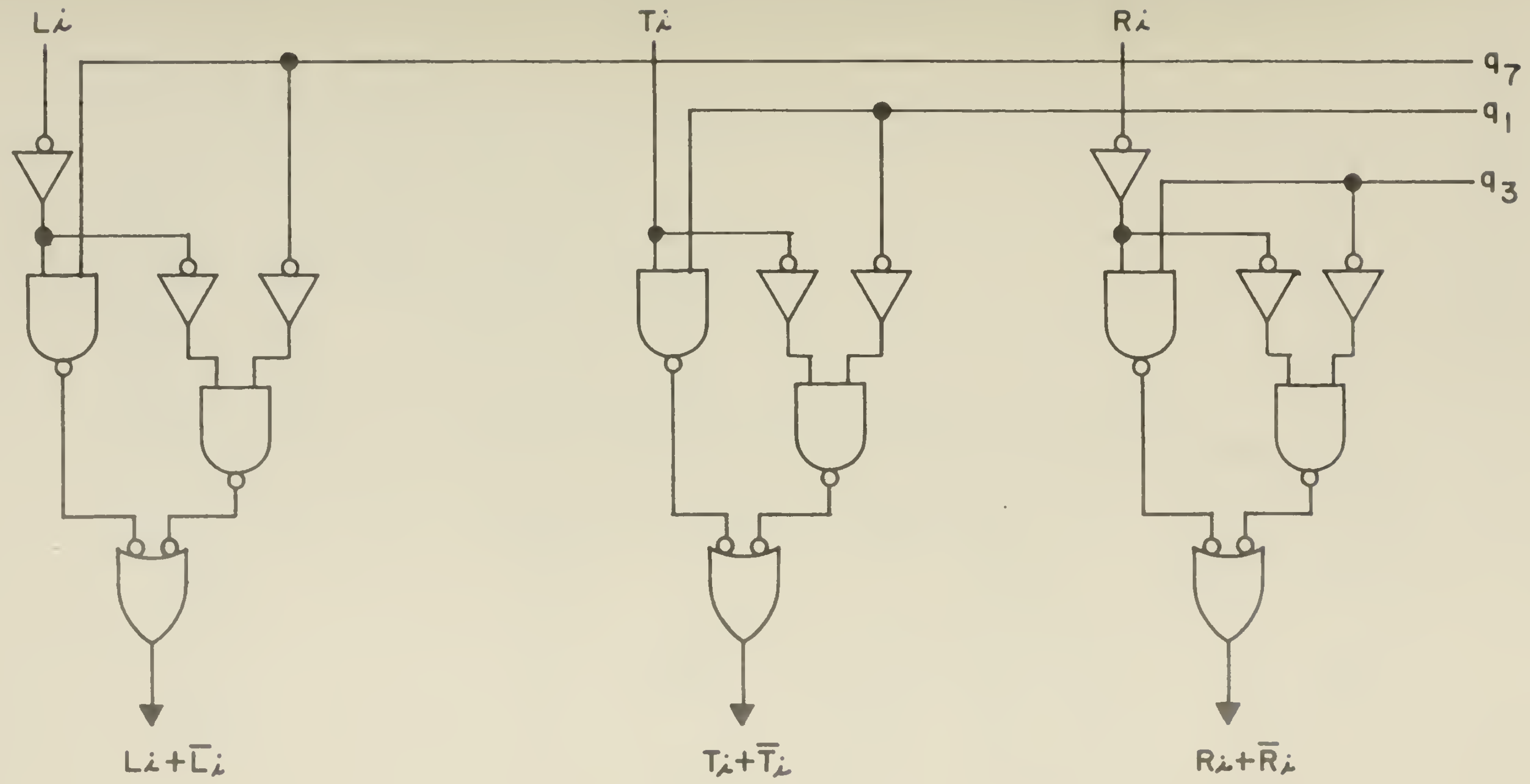


FIGURE 3. SQUARE CELL FUNCTION OR COMPLEMENT DECISION LOGIC.

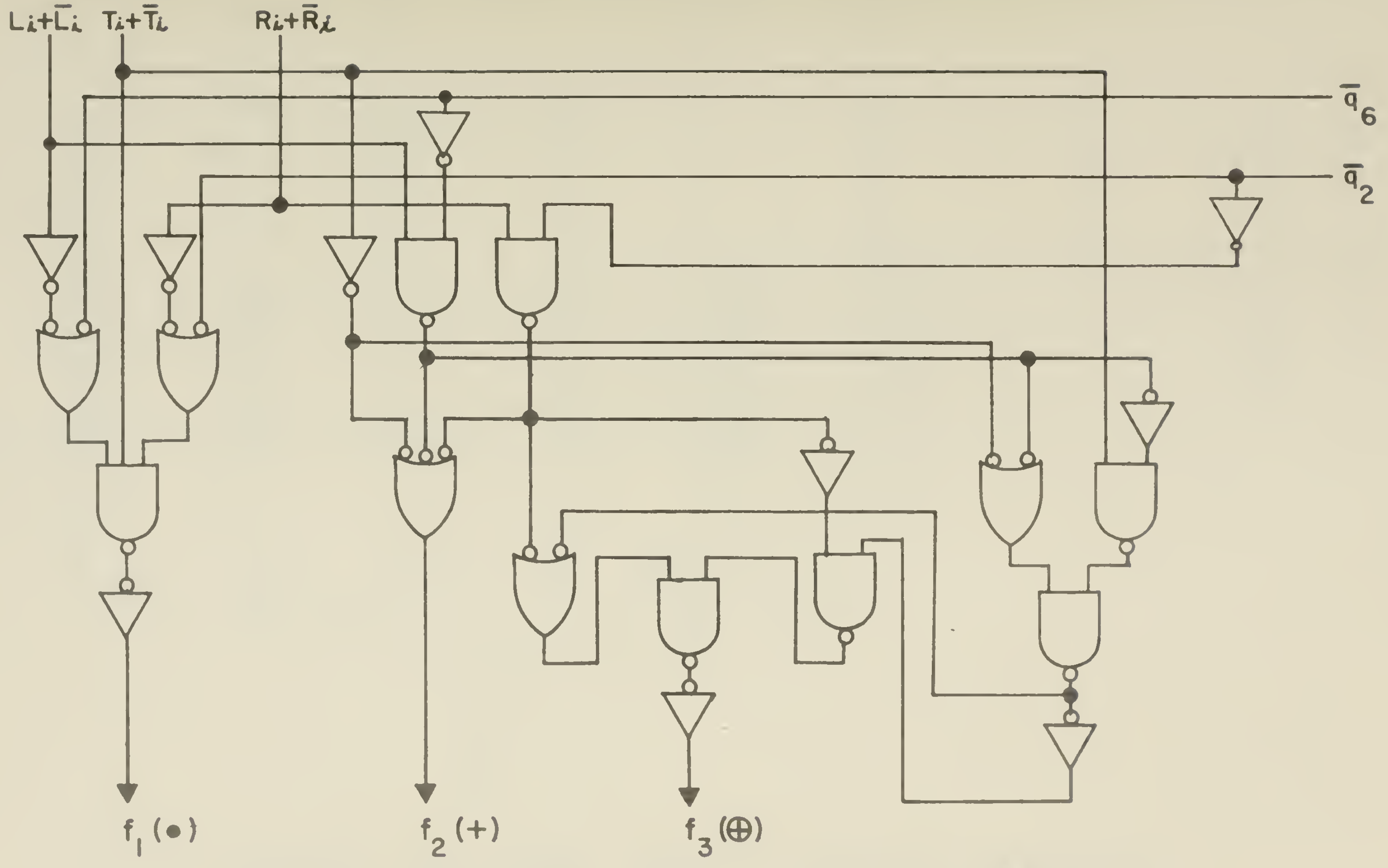


FIGURE 4. SQUARE CELL BASIC FUNCTIONS GENERATOR.

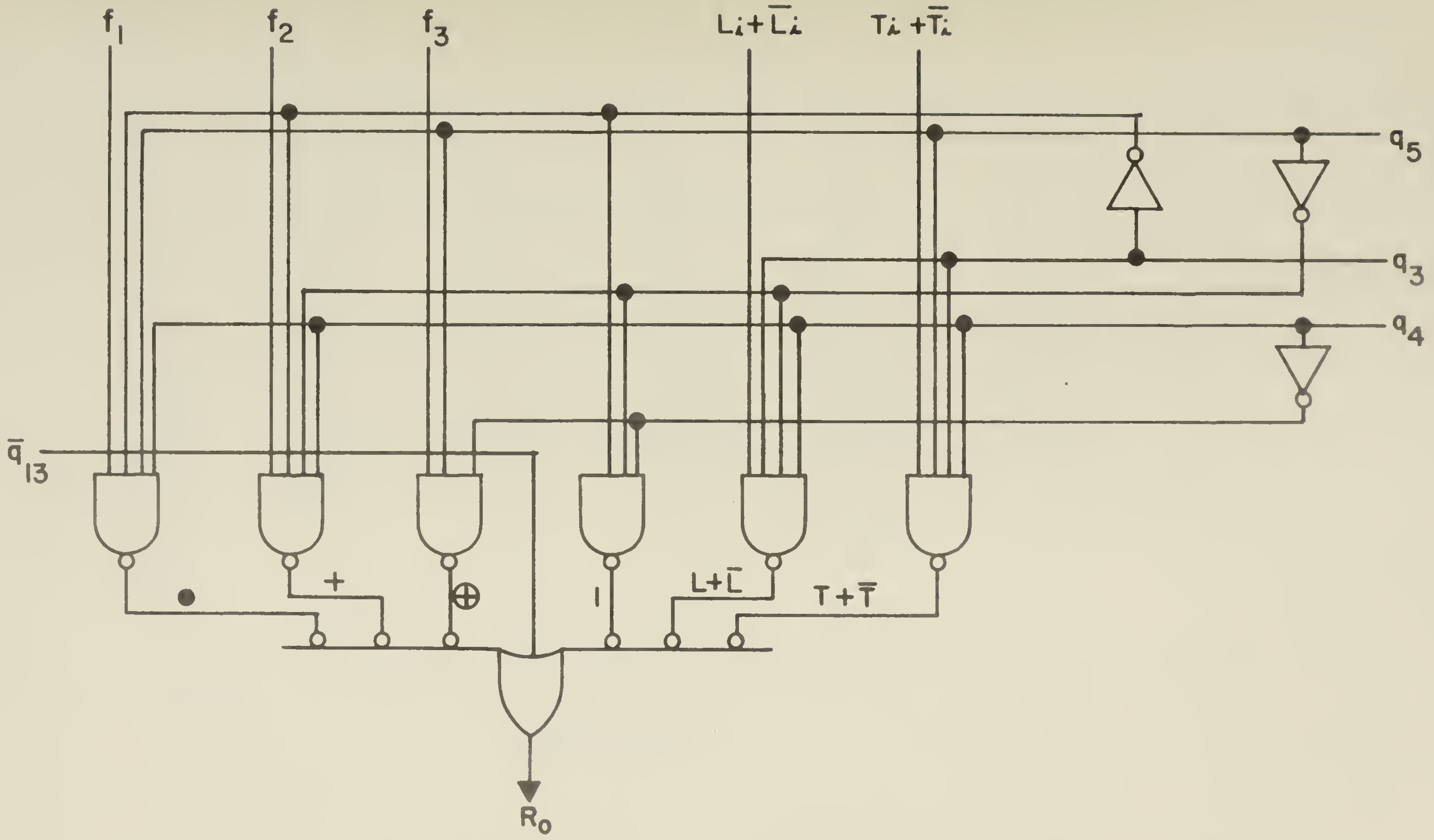


FIGURE 5. SQUARE CELL RIGHT OUTPUT FUNCTION SELECTION.

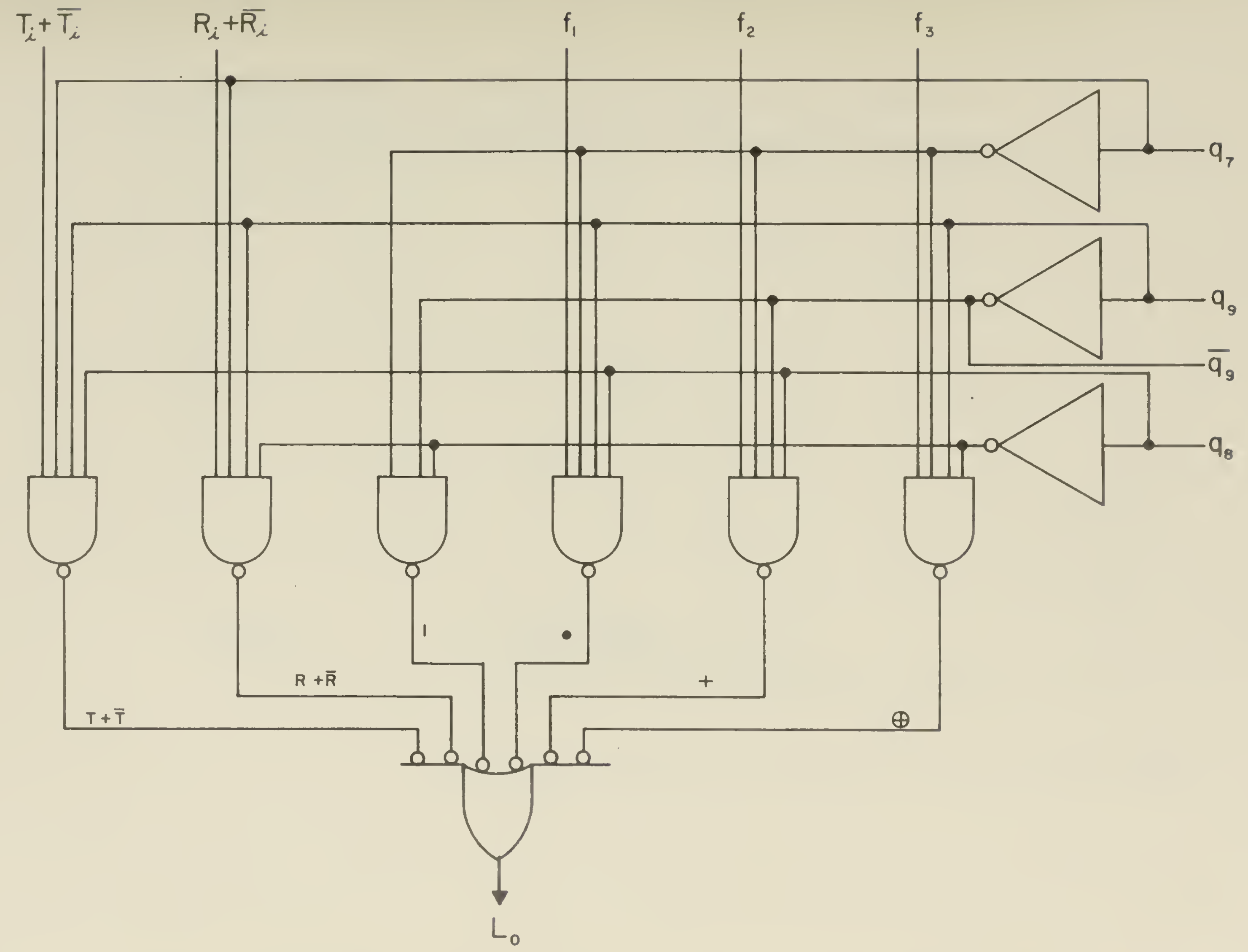
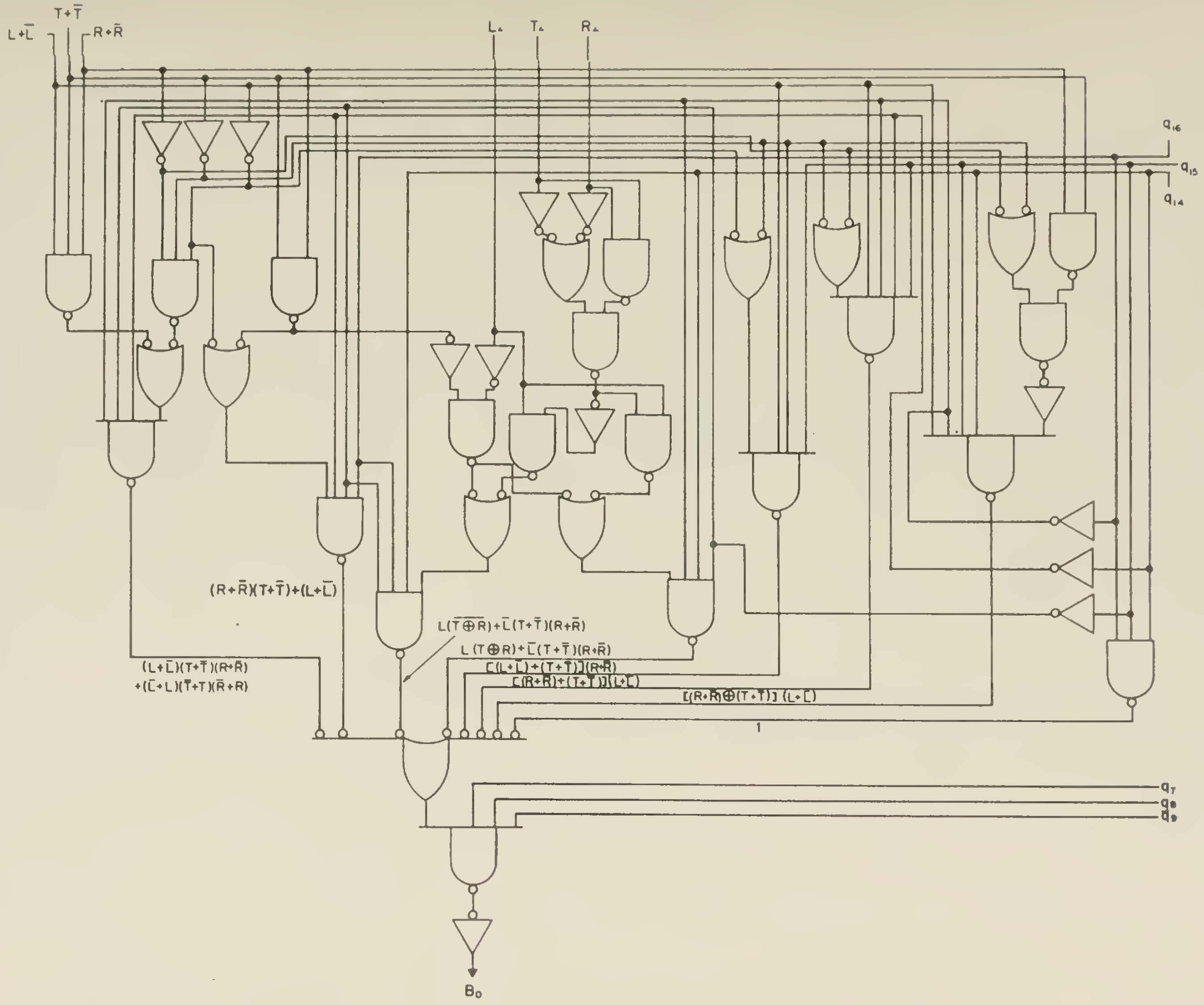


FIGURE 6. SQUARE CELL LEFT OUTPUT FUNCTION SELECTION.



$L+\bar{L}$
$T+\bar{T}$
$R+\bar{R}$
$L_\perp$
$T_\perp$
$R_\perp$
q_{16}
q_{15}
q_{14}
$(R+\bar{R})(T+\bar{T})+(L+\bar{L})$
$L(\overline{T \oplus R})+\bar{L}(T+\bar{T})(R+\bar{R})$
$L(T \oplus R)+\bar{L}(T+\bar{T})(R+\bar{R})$
$[(L+\bar{L})+(T+\bar{T})](R+\bar{R})$
$[(R+\bar{R})+(T+\bar{T})](L+\bar{L})$
$[(R+\bar{R})\oplus(T+\bar{T})](L+\bar{L})$
$(L+\bar{L})(T+\bar{T})(R+\bar{R})$
$+(\bar{L}+L)(\bar{T}+T)(\bar{R}+R)$
1
q_7
q_8
q_9
B_0

FIGURE 7. SQUARE CELL BOTTOM OUTPUT SPECIAL FUNCTIONS GENERATION AND SELECTION.

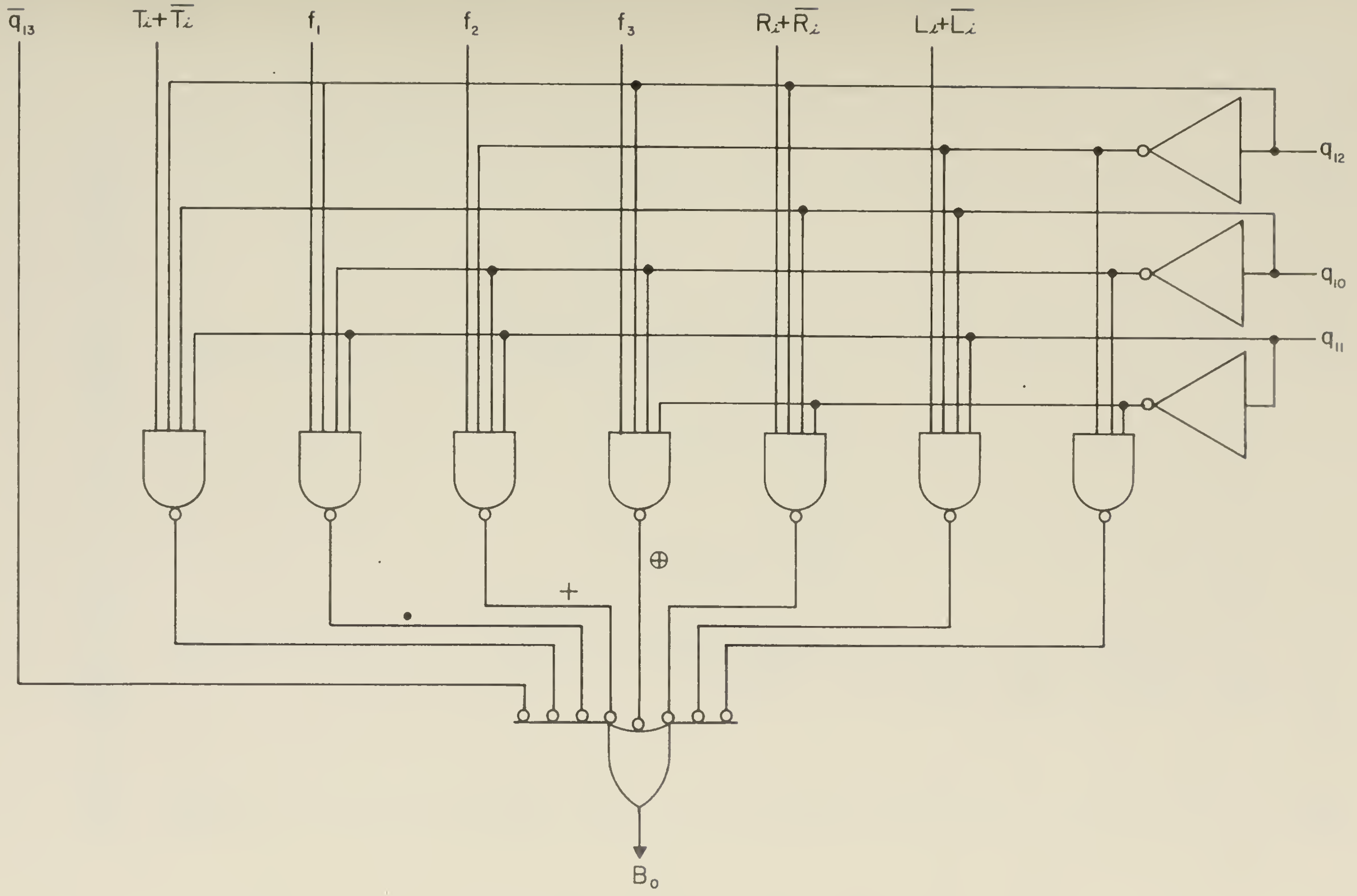


FIGURE 8. SQUARE CELL BOTTOM OUTPUT BASIC FUNCTIONS SELECTION.

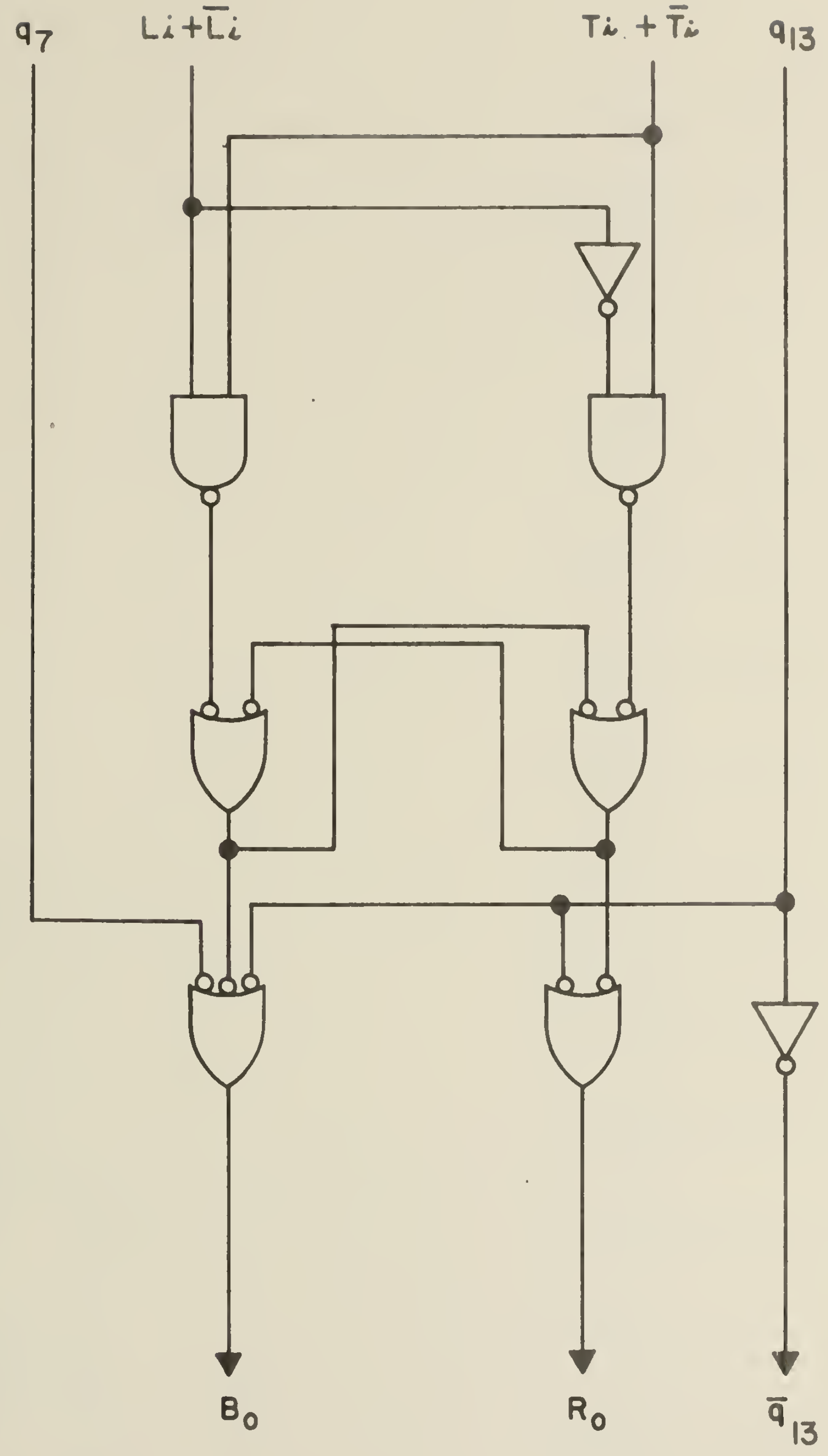


FIGURE 9. SQUARE CELL FLIP-FLOP LOGIC

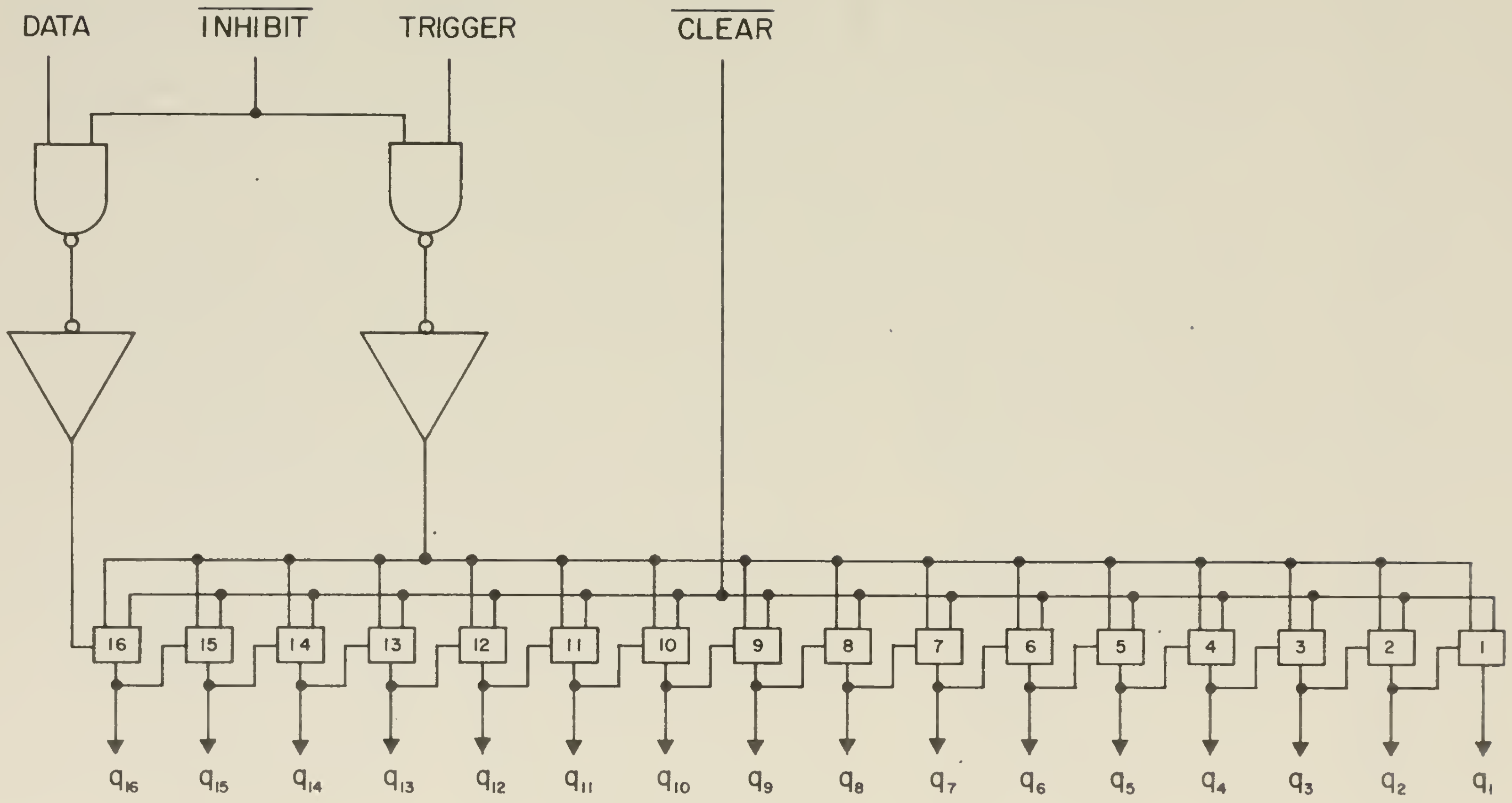


FIGURE 10. ADDRESS LOGIC

and the 16 bits would be shifted in by the continuously running trigger. The $\overline{\text{Clear}}$ line sets the shift register of Figure 11 back to all zeros.

Of course, it is not essential that the square cell, as well as the hexagonal cell to be discussed in the next section, have the internal programming depicted in the above paragraph. This was only one of several possible methods. Another is that a register be employed outside the cells in a separate unit. In fact, individual lines to each cell could even replace the registers. Actually, the number of cells used in the computer elements would probably dictate the choice of the type of cell programming.

3.4 Advantages and Disadvantages

The advantage of the square cell over the cutpoint cell is that many more functions can be formed in one cell. Only 8 of the 16 functions of two variables can be performed using a cutpoint cell while with one square cell, 152 of the 256 functions of three variables can be constructed. Appendix B gives a listing of the functions that can be made with one cell and those that take two cells. Included in those that need only one cell are, of course, all of the functions of two variables. Thus, the many more functions that this cell provides results in fewer cells used in digital circuits.

The saving in the number of cells used is obtained by simplifying the function involved. This, however, yields no simple scheme to program the cells. For a programmer, though, the job of setting up the cell pattern and programming them does not appear to be too difficult. Also, further investigation could probably yield a method in which a computer would be utilized to determine the individual cell settings.

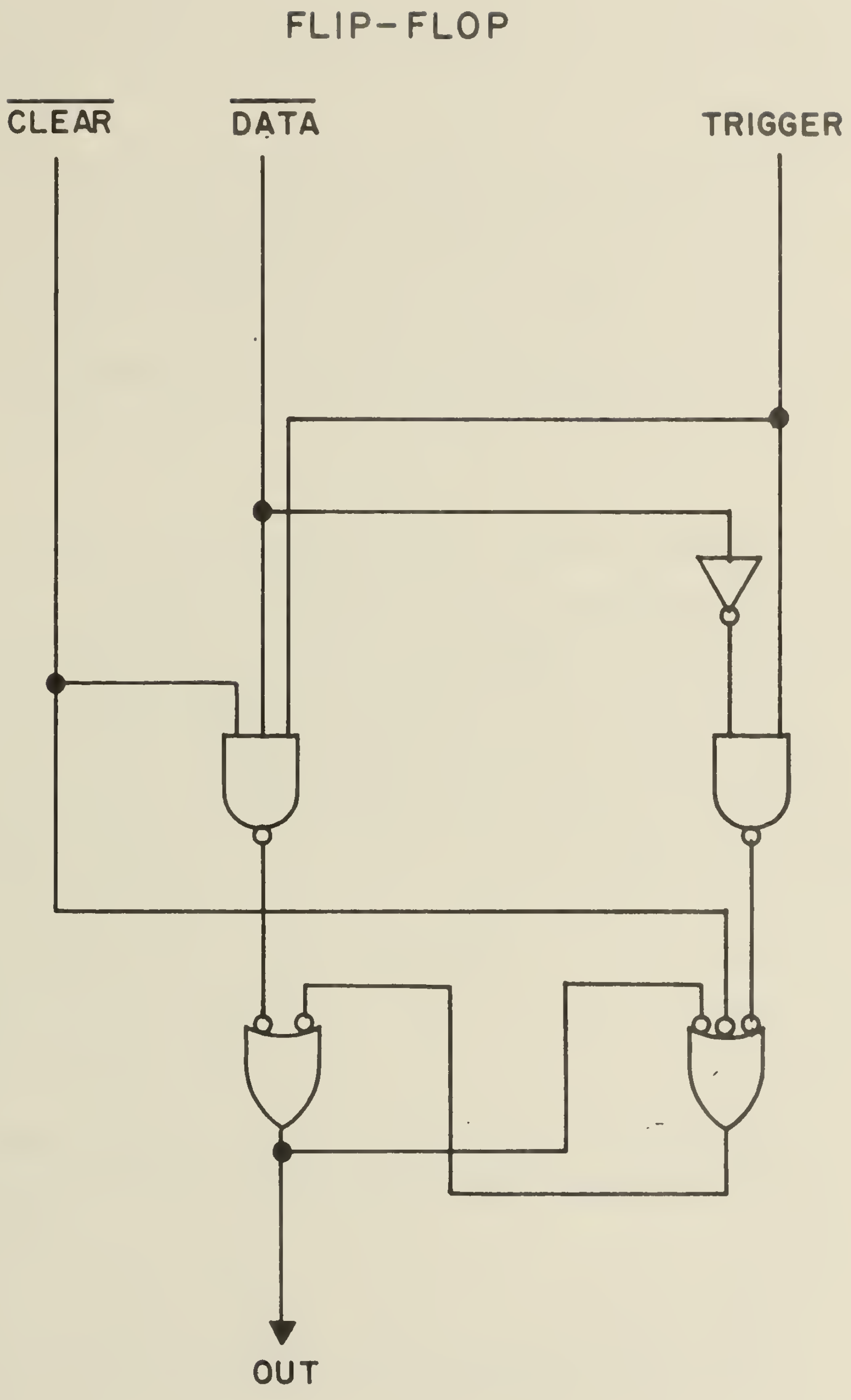


FIGURE 11. ADDRESS LOGIC FLIP-FLOP.

4. HEXAGONAL CELL

4.1 Cell Design

The hexagonal cell's block diagram is shown in Figure 12. The address elements are the same as for the square cell and can be found by referring back to Figures 1, 10 and 11. There are three fixed inputs and three fixed outputs associated with this type of cell. One of the outputs is the same or the complement of one of the inputs while both the D and E output circuits are identical.

4.2 Coding

The coding for the hexagonal cell is listed in Table 3.

The first decision is whether or not to complement the inputs and then setting bits q_1, q_2 and q_3 accordingly. In forming the functions involved, anywhere from none to all of the inputs may be needed. q_4, q_5 and q_6 are used to inhibit the inputs used for the D output while q_{10}, q_{11} and q_{12} are used for the E output. Cell functions are selected using bits 7 to 9 for the D output and 13 to 15 for the E output. The functions

$$(A+\bar{A})\ (B+\bar{B})\ (C+\bar{C})\ +\ (\bar{A}+A)\ (\bar{B}+B)\ (\bar{C}+C) \qquad (4.1)$$

$$A(\overline{B\oplus C})\ +\ \bar{A}(B+\bar{B})\ (C+\bar{C}) \qquad (4.2)$$

and

$$A(B\oplus C)\ +\ \bar{A}(B+\bar{B})\ (C+\bar{C}) \qquad (4.3)$$

which are the same as functions 3.1, 3.2 and 3.3, are included to avoid the necessity of using three cells to obtain these functions. The fourth function

$$((A+\bar{A})\ +\ (B+\bar{B}))\ (C+\bar{C}) \qquad (4.4)$$

was picked since it is the representative function of one of the

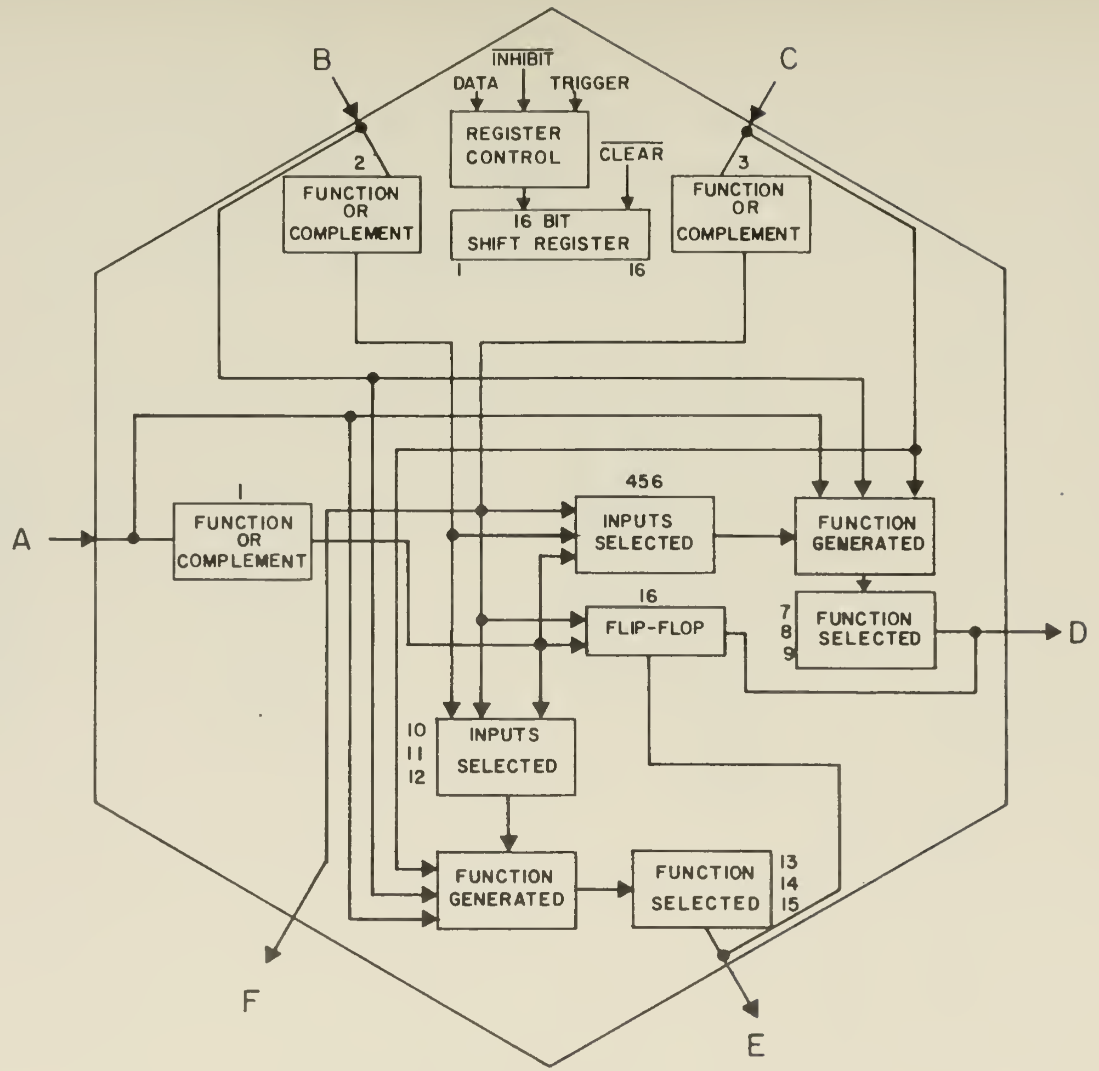


FIGURE 12. HEXAGONAL CELL BLOCK DIAGRAM.

TABLE 3: HEXAGONAL CELL CONTROLS

q_1:	0	$\overline{A}$	q_2: 0	$\overline{B}$	q_3:	0	$\overline{C}$
	1	A	1	B		1	C

D OUTPUTS			E OUTPUTS
q_4: 0	A INPUT INHIBITED		q_{10}: 0
q_5: 0	B " "		q_{11}: 0
q_6: 0	C " "		q_{12}: 0

q_4: 0	q_5: 0	q_6: 0	q_7: 0	0	q_{10}: 0	q_{11}: 0	q_{12}: 0	q_{13}: 0
			1	1				1

q_7	q_8	q_9		q_{13}	q_{14}	q_{15}
0	0	1	$\cdot$	0	0	1
0	1	0	$+$	0	1	0
0	1	1	$\oplus$	0	1	1
1	0	0	$(A+\overline{A})(B+\overline{B})(C+\overline{C})+(\overline{A}+A)(\overline{B}+B)(\overline{C}+C)$	1	0	0
1	0	1	$[(A+\overline{A})+(B+\overline{B})]\ (C+\overline{C})$	1	0	1
1	1	0	$A(\overline{B\oplus C})+\overline{A}(B+\overline{B})(C+\overline{C})$	1	1	0
1	1	1	$A(B\oplus C)+\overline{A}(B+\overline{B})(C+\overline{C})$	1	1	1

q_{16}: 1 FLIP-FLOP

	$A+\overline{A}$	$B+\overline{B}$	$C+\overline{C}$	E	D
q_9: 1	1	1	d	0	1
	0	1	d	1	0
	1	0	d	OUTPUTS SAME AS PREVIOUS OUTPUTS	
	0	0	d		
q_9: 0	0	1	0		
	0	1	1	1	0
	1	1	0	0	1
	1	1	1	0	0
		0		OUTPUTS SAME AS PREVIOUS OUTPUTS	

classes with the maximum number of members (24). The flip-flop is a Set-Reset flip-flop and is controlled by q_9 and q_{16}. When this type of operation is desired, D and E are the outputs while A is the set, C is the reset, and B is the trigger.

4.3 Logic Implementation

The implementation of the hexagonal cell is similar to that of the square cell. The function or complement decision logic is shown in Figure 13 while the D functions are formed and generated in Figures 14 and 15. The E functions are formed in exactly the same way except that the control bits are different. A Set-Reset flip-flop is included and illustrated in Figure 16. Finally, the address logic used is the same as that for the square cell and can be found by referring back to Figures 10 and 11.

4.4 Advantages and Disadvantages

As for the square cell, the hexagonal cell can also produce all 256 functions of three variables using two cells. Of these, only 92 can be formed with one cell and would therefore indicate that a square cell would be the better cell. As the examples of Chapter 5 illustrate, however, the hexagonal cell usage results in the minimum number of cells. This is due to the fact that there are always three outputs and three inputs with the hexagonal cell. Thus, hexagonal cells, not requiring an input/output decision, have a more logical coding scheme and are easier to program.

The hexagon, as well as the square, have the disadvantage that if a variable is used in both output functions, either the variable or the complement is used but not both. This can be corrected in many instances by complementing the input of the adjoining cell rather than adding additional gates in each cell.

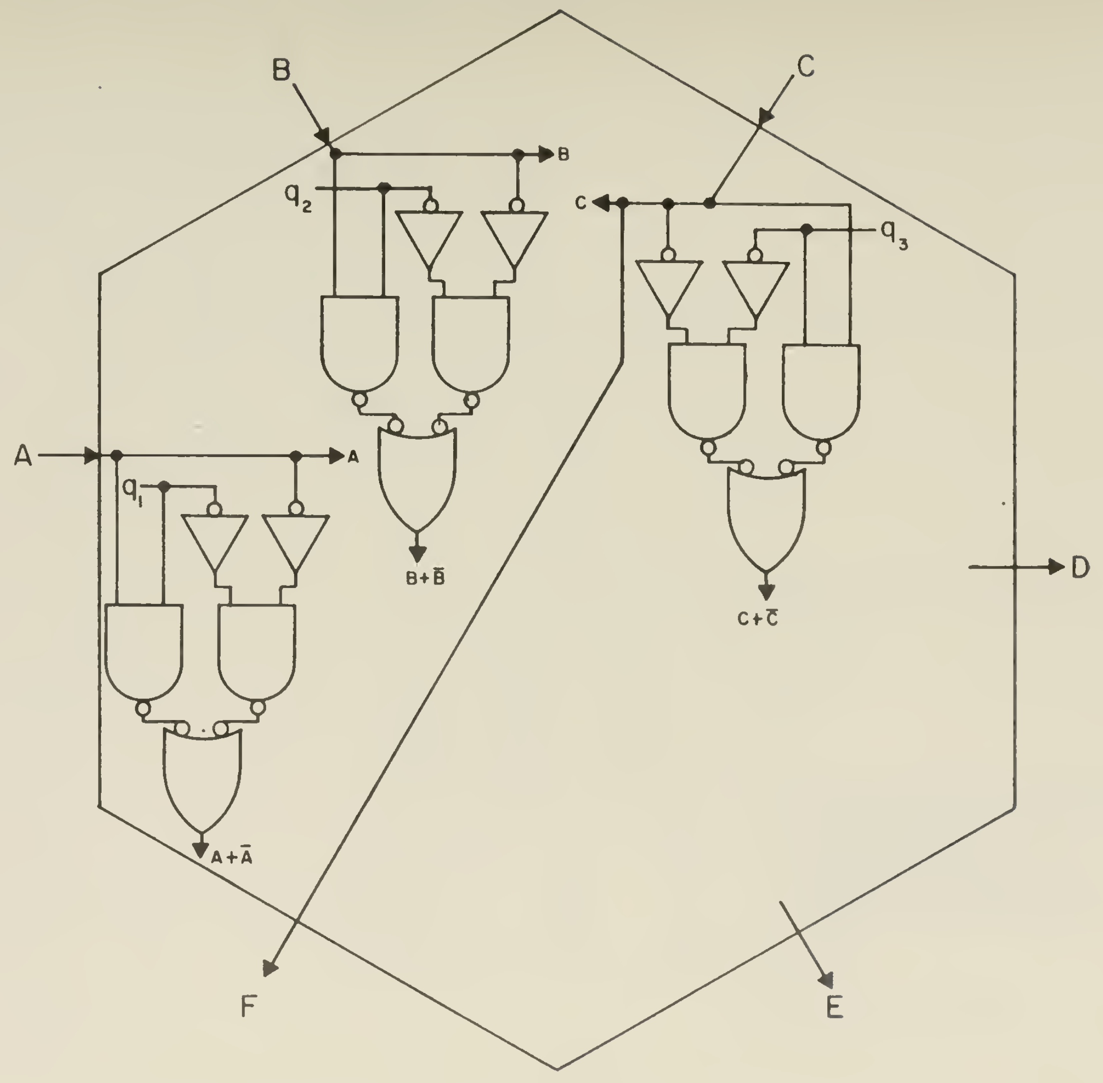


FIGURE 13. HEXAGONAL CELL FUNCTION OR COMPLEMENT DECISION LOGIC.

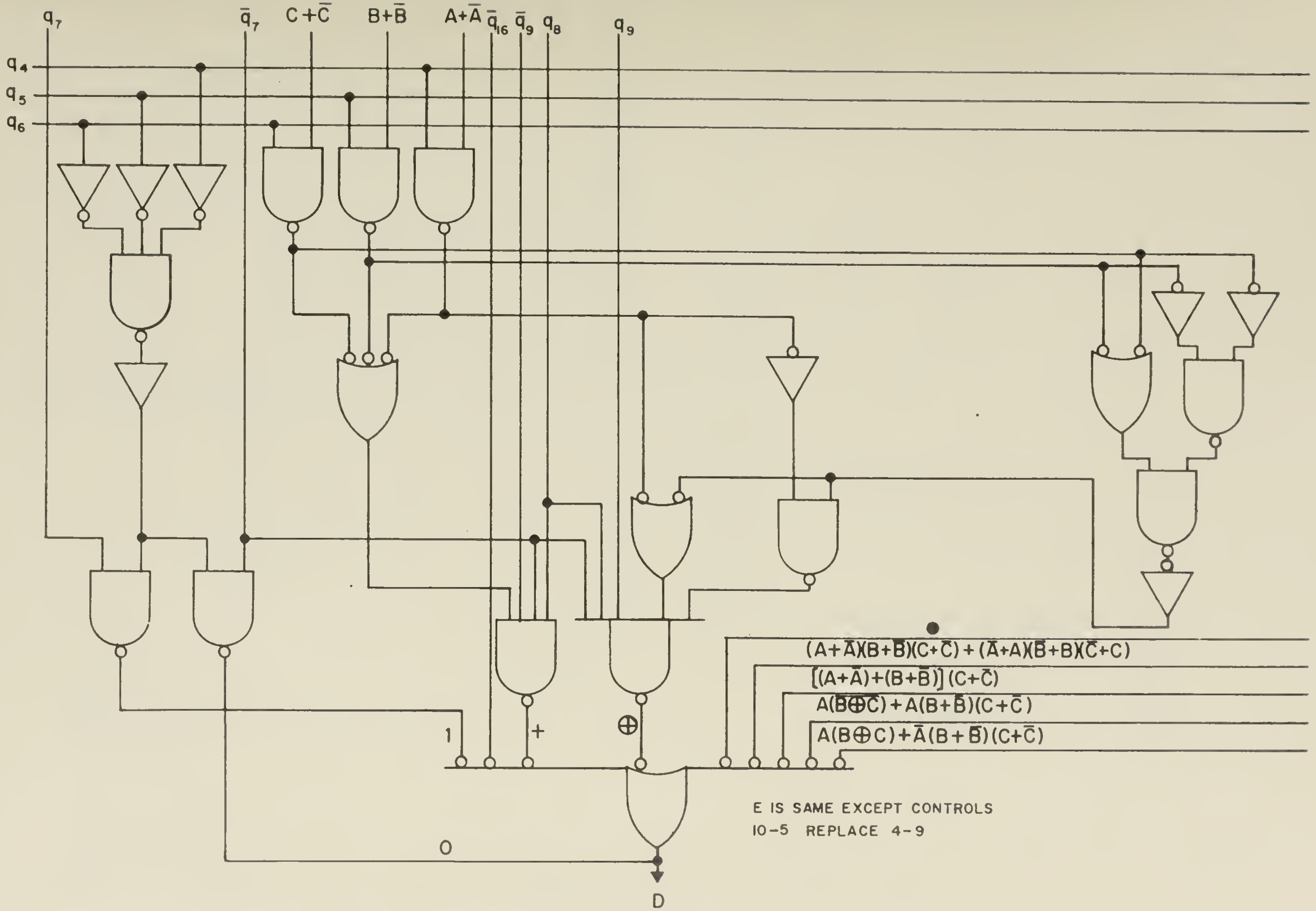

q_7
$\bar{q}_7$
$C+\bar{C}$
$B+\bar{B}$
$A+\bar{A}$
$\bar{q}_{16}$
$\bar{q}_9$
q_8
q_9
q_4
q_5
q_6
$(A+\bar{A})(B+\bar{B})(C+\bar{C})+(\bar{A}+A)(\bar{B}+B)(\bar{C}+C)$
$[(A+\bar{A})+(B+\bar{B})](C+\bar{C})$
$A(\bar{B}\oplus\bar{C})+A(B+\bar{B})(C+\bar{C})$
$A(B\oplus C)+\bar{A}(B+\bar{B})(C+\bar{C})$
1
+
$\oplus$
0
D
E IS SAME EXCEPT CONTROLS
10-5 REPLACE 4-9

FIGURE 14. HEXAGONAL CELL FUNCTION GENERATION AND SELECTION PART a

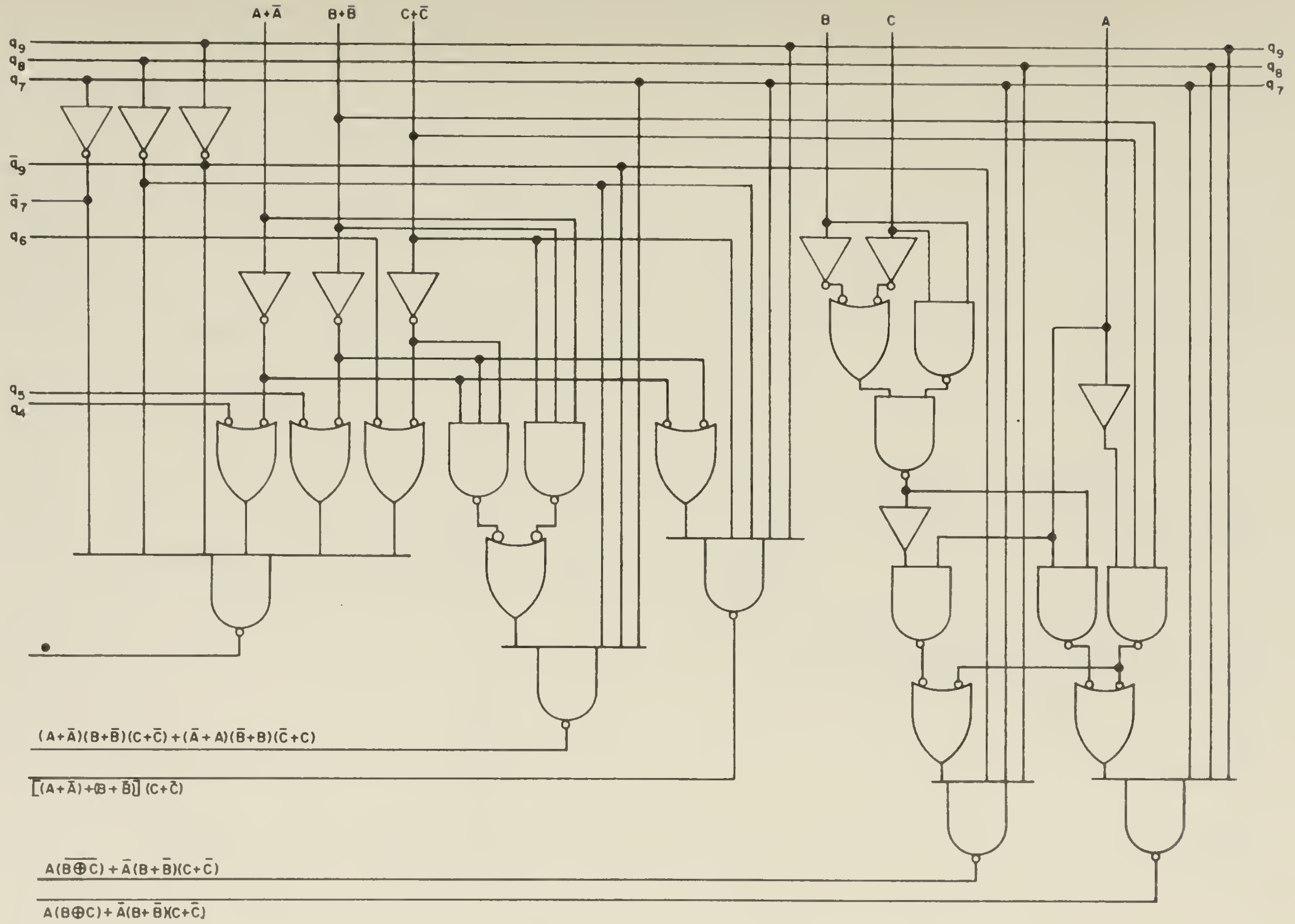


FIGURE 15. HEXAGONAL CELL FUNCTION GENERATION AND SELECTION PART b

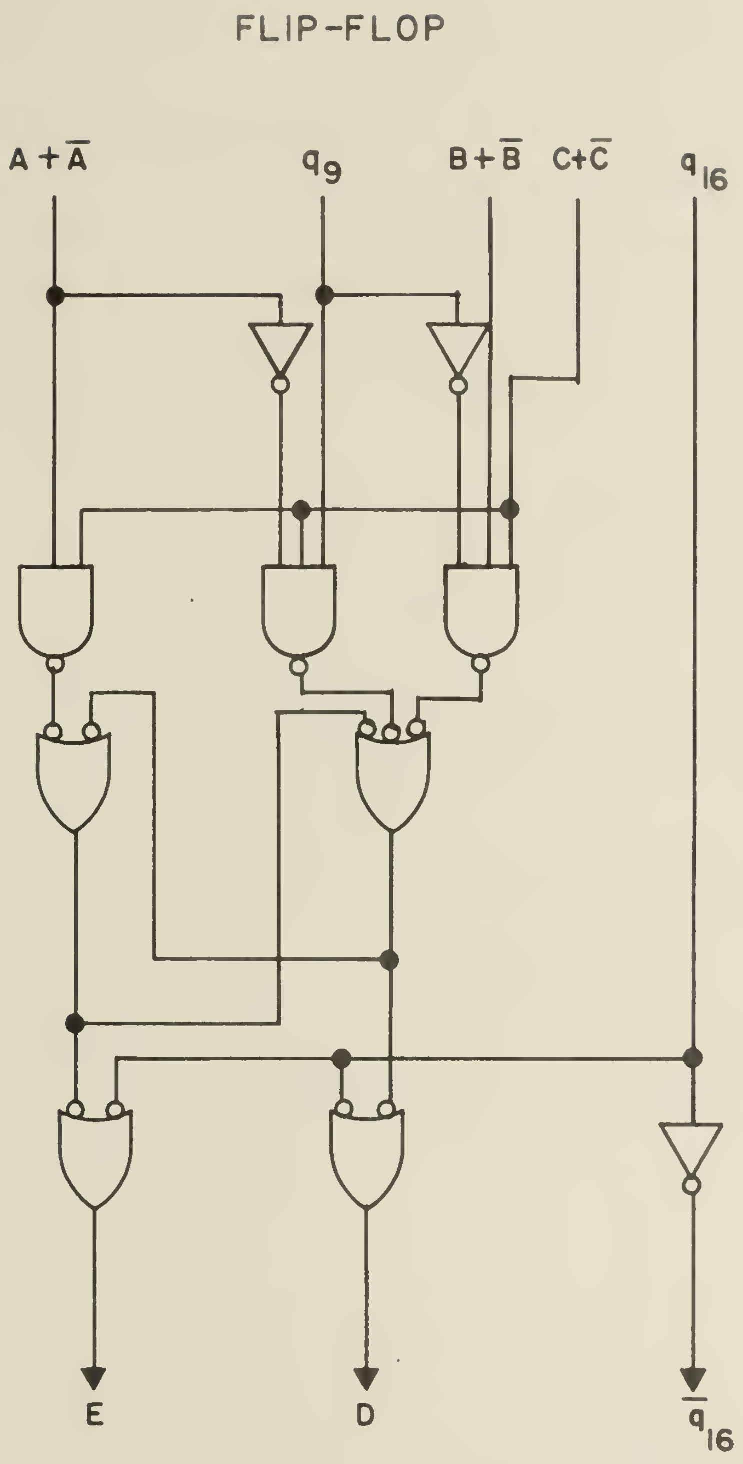


FIGURE 16. HEXAGONAL CELL FLIP-FLOP LOGIC

5. EXAMPLES OF THE USE OF CELLULAR LOGIC

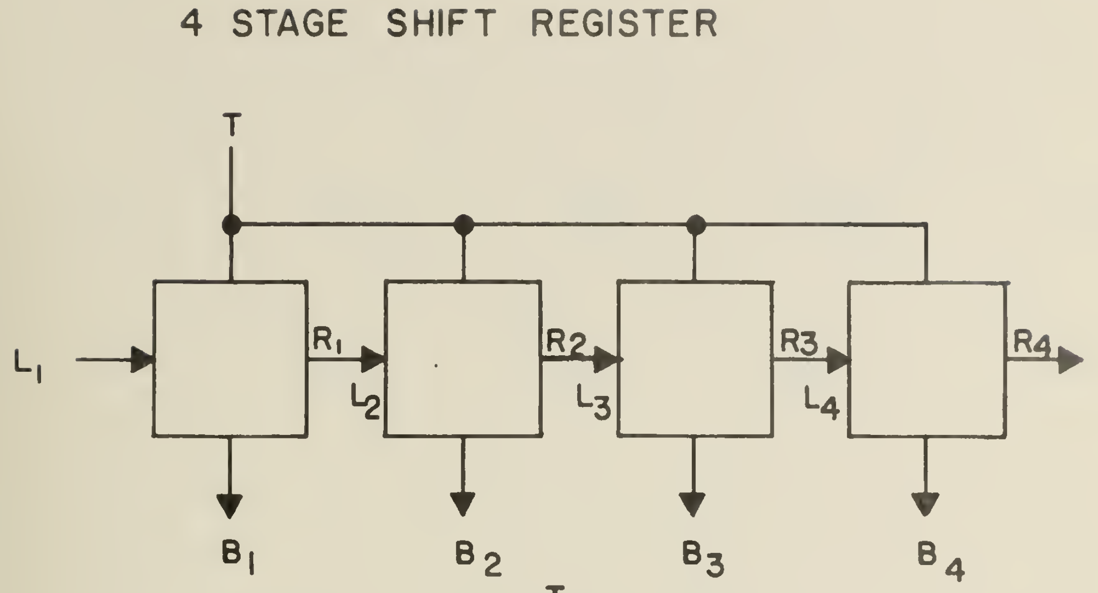
Since each of the three cells discussed in this thesis contain a flip-flop, binary counters and shift registers can be made by the proper combination of cells. Figures 17 and 18 show that the square method and the hexagon method use the same number of cells for a shift register while in Figure 19, the cutpoint method requires four times the number of cells.

Another common digital circuit is a three—variable decoder. Figures 20, 21 and 22, which are examples of this type of circuit, show that there are twice as many square cells and three times as many cutpoint cells used than hexagonal cells.

Figure 23 shows a nines complement circuit and the individual cell coding for the square cell method. Figures 24 and 25 are the hexagonal and cutpoint methods respectively.

A translator from binary-coded decimal to two-out-of-five circuit is shown in the next three figures (26, 27 and 28). As usual, the hexagonal array requires the least number of cells.

Further examples of using cellular logic to form functions that are used in digital work are shown in Figures 29 through 37. As before, the cutpoint arrays use the most cells in every case while the hexagonal method uses the least.



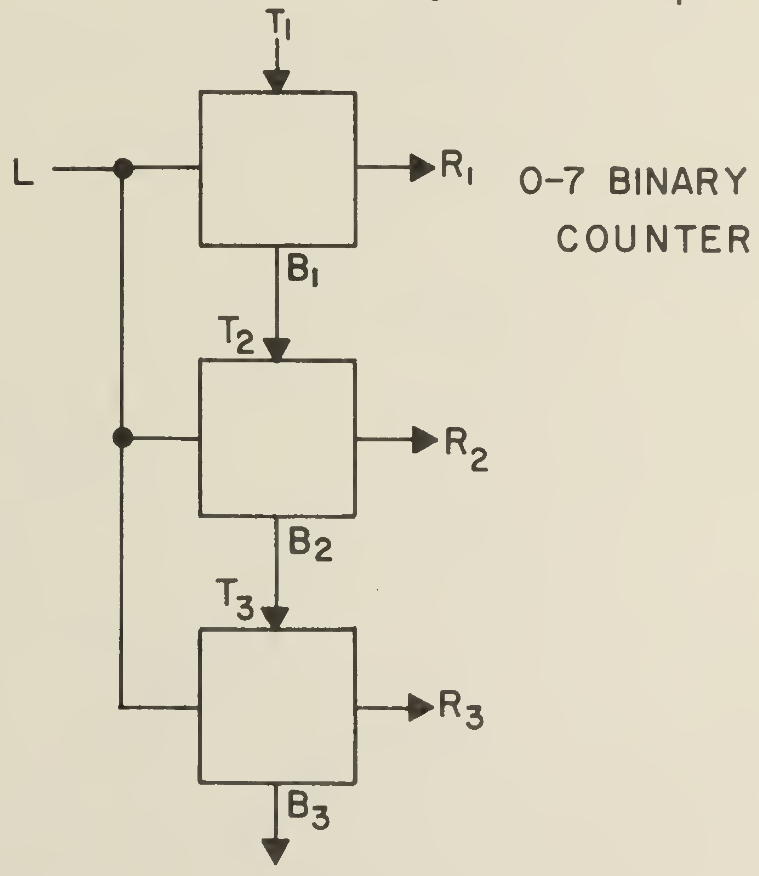
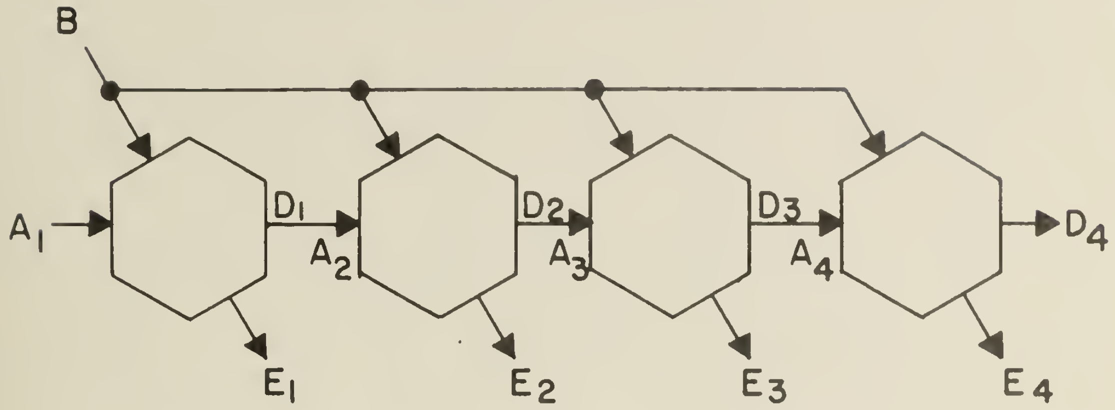


FIGURE 17. SQUARE CELL REGISTER AND COUNTER.

4 STAGE SHIFT REGISTER



0-7 BINARY COUNTER

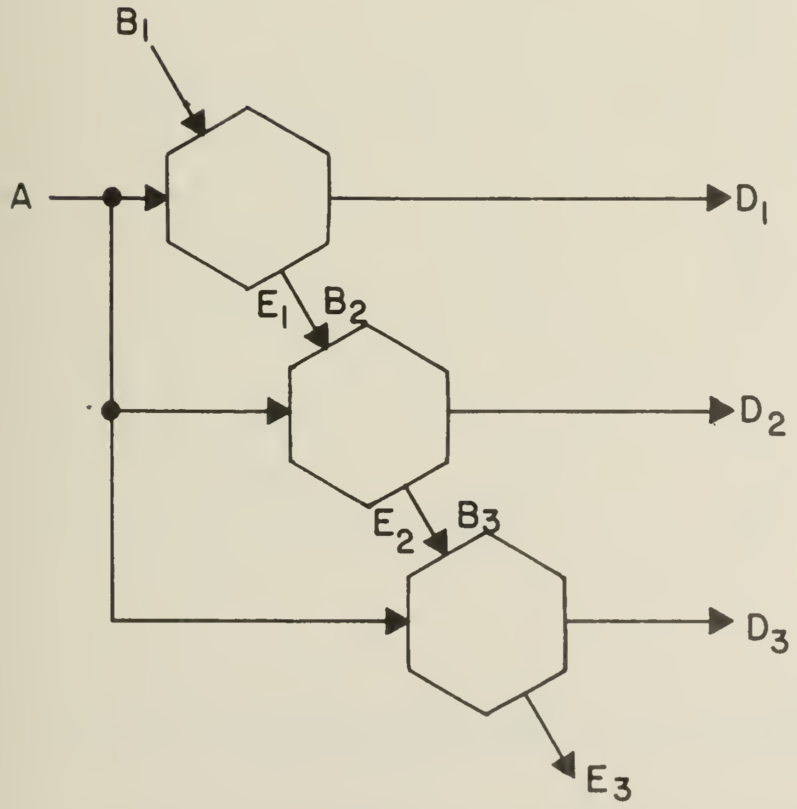


FIGURE 18. HEXAGONAL CELL REGISTER AND COUNTER.

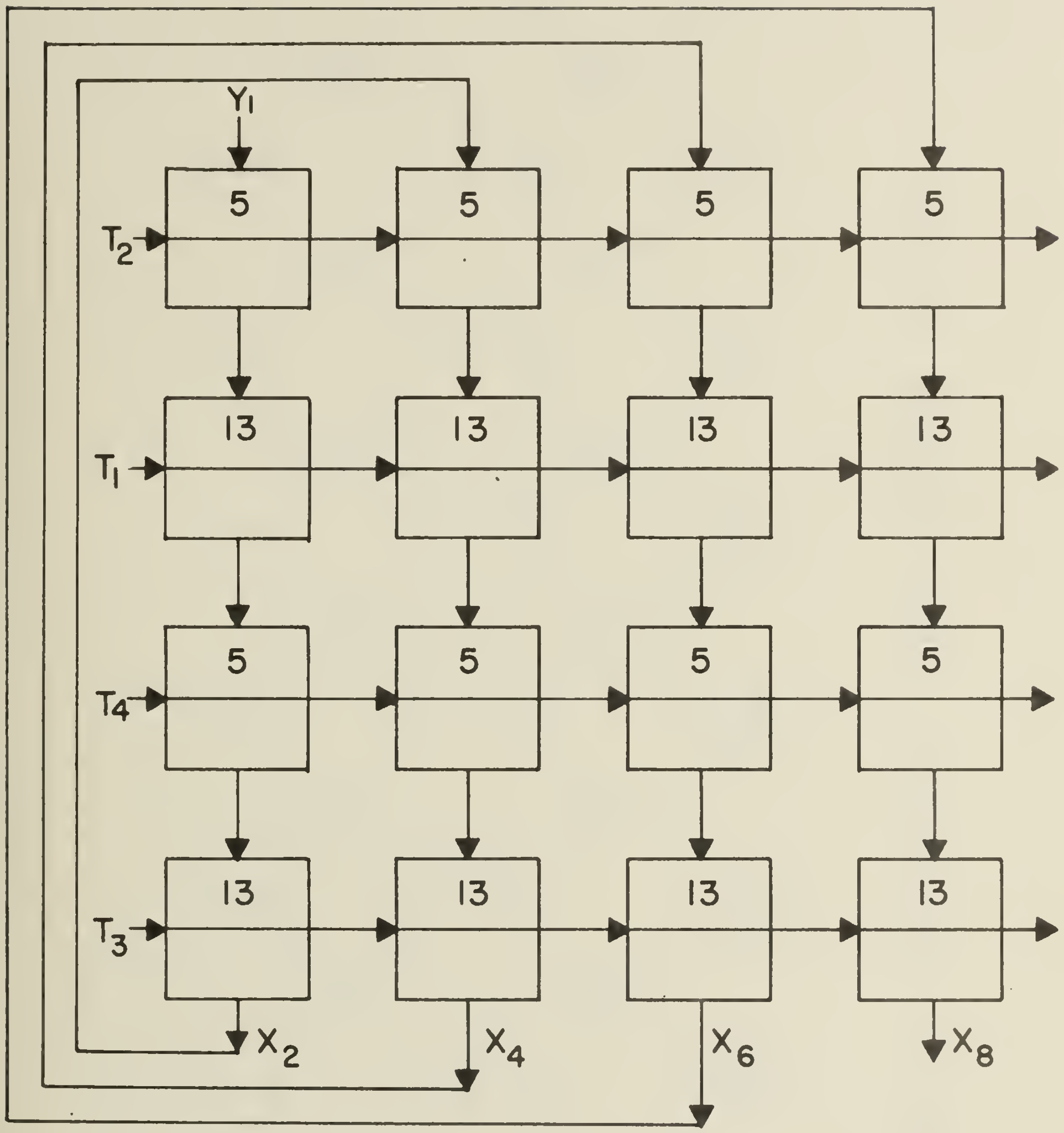


FIGURE 19. CUTPOINT CELL FOUR STAGE SHIFT REGISTER.*

*Minnick, R.C., "Cutpoint Cellular Logic," p.696

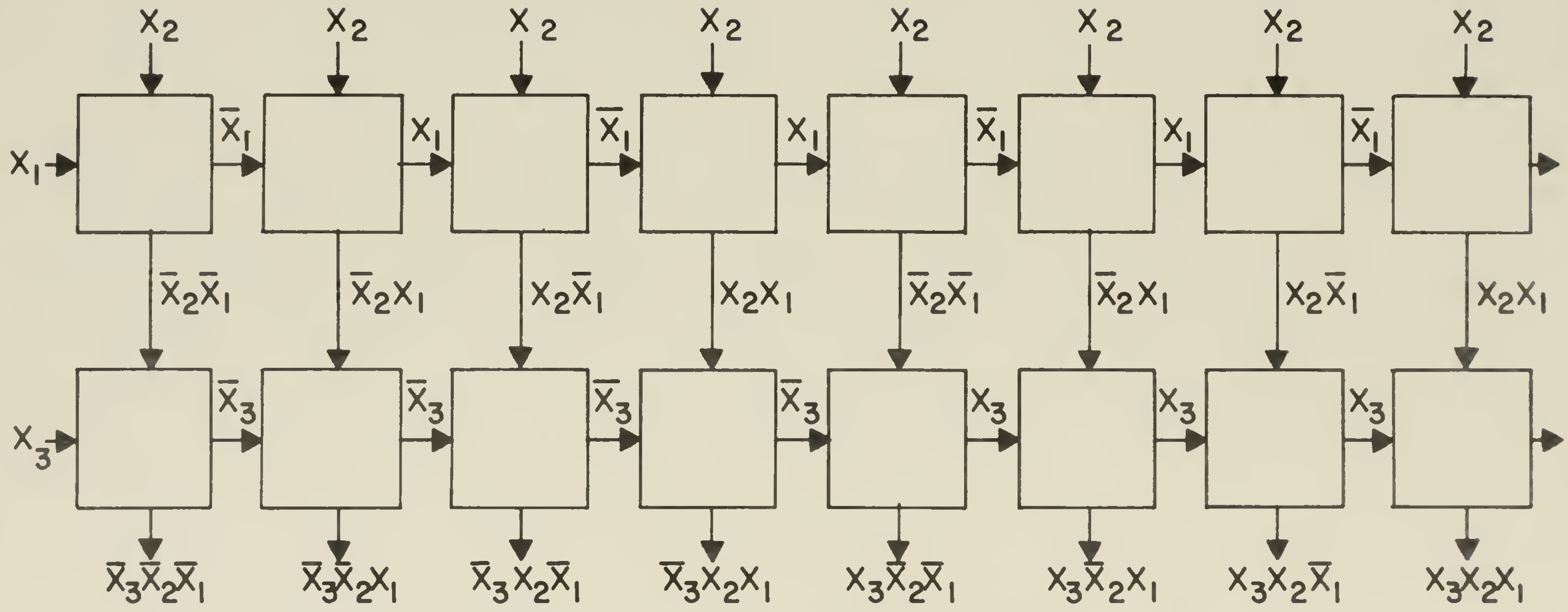

X_2
X_1
$\overline{X}_1$
X_3
$\overline{X}_3$
$\overline{X}_2\overline{X}_1$
$\overline{X}_2X_1$
$X_2\overline{X}_1$
X_2X_1
$\overline{X}_3\overline{X}_2\overline{X}_1$
$\overline{X}_3\overline{X}_2X_1$
$\overline{X}_3X_2\overline{X}_1$
$\overline{X}_3X_2X_1$
$X_3\overline{X}_2\overline{X}_1$
$X_3\overline{X}_2X_1$
$X_3X_2\overline{X}_1$
$X_3X_2X_1$

FIGURE 20. SQUARE CELL THREE VARIABLE DECODER.

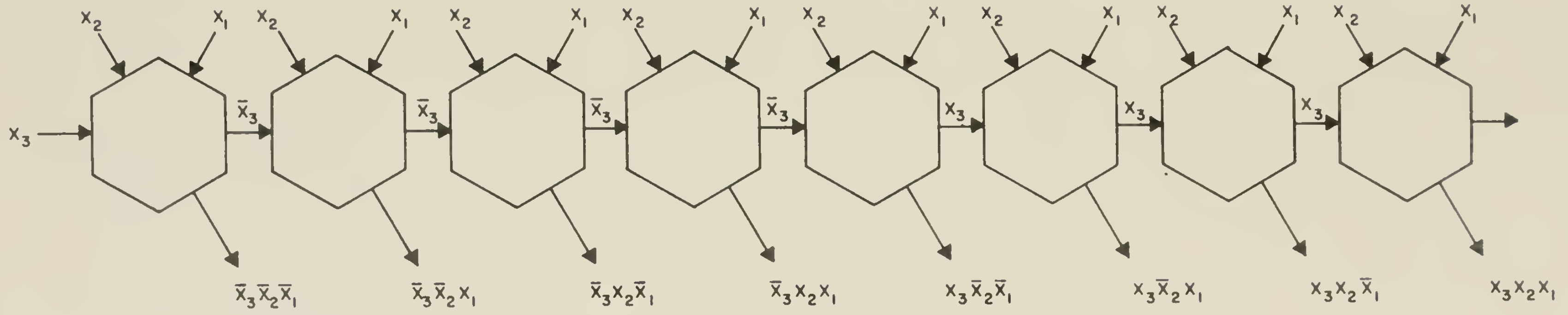


FIGURE 21. HEXAGONAL CELL THREE VARIABLE DECODER.

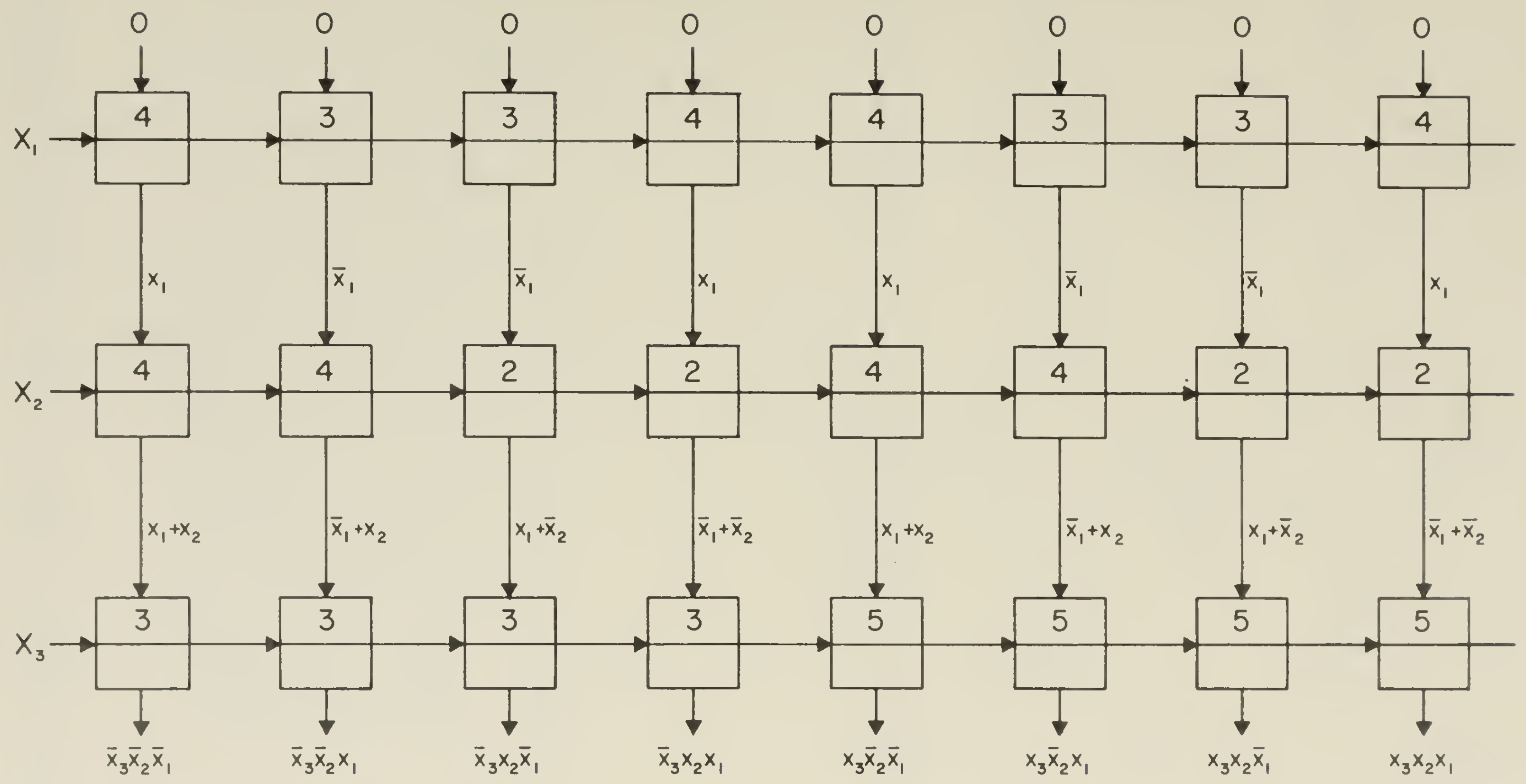
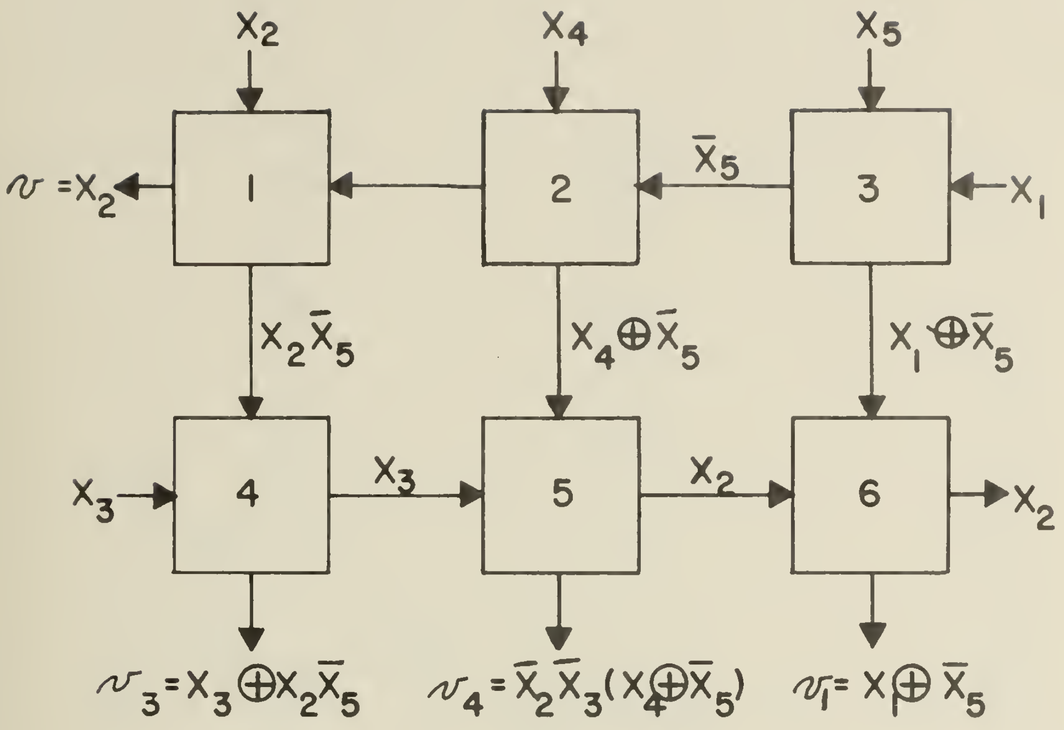


FIGURE 22. CUTPOINT CELL THREE VARIABLE DECODER.*

* Minnick, R.C., "Cutpoint Cellular Logic," p.694



CODE / CELL	1	2	3	4	5	6	7	8	9	10	11	12	13	14-16
1	1	1	1	d	d	0	1	1	1	1	1	0	0	1
2	1	1	1	d	d	0	1	0	1	0	1	0	0	1
3	0	1	1	d	d	0	1	1	1	0	1	0	0	1
4	1	0	1	1	0	1	1	d	d	0	1	0	0	1
5	1	1	0	d	d	1	0	d	d	1	1	0	0	1
6	1	1	1	d	d	0	1	0	1	1	1	1	0	1

FIGURE 23. SQUARE CELL NINES COMPLEMENT CIRCUIT

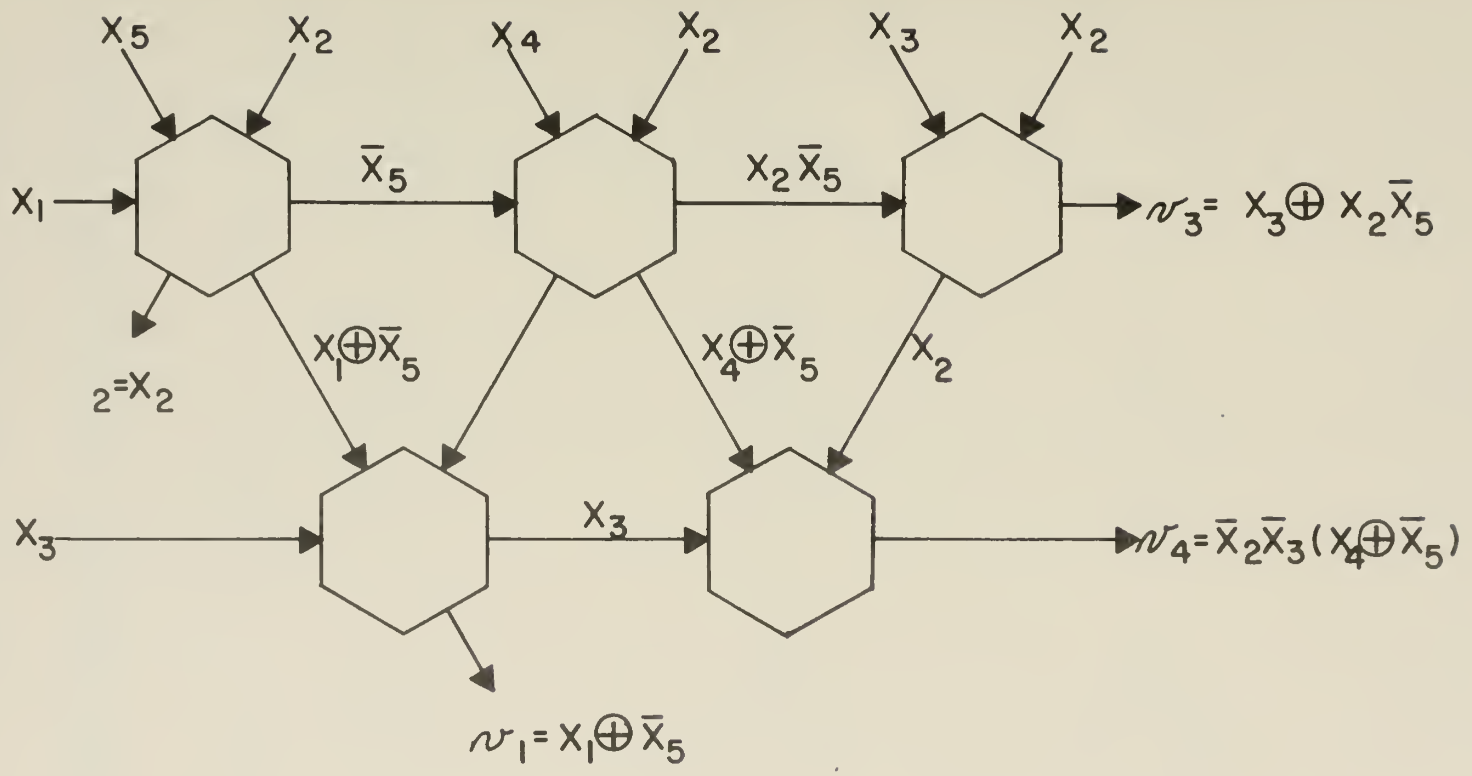
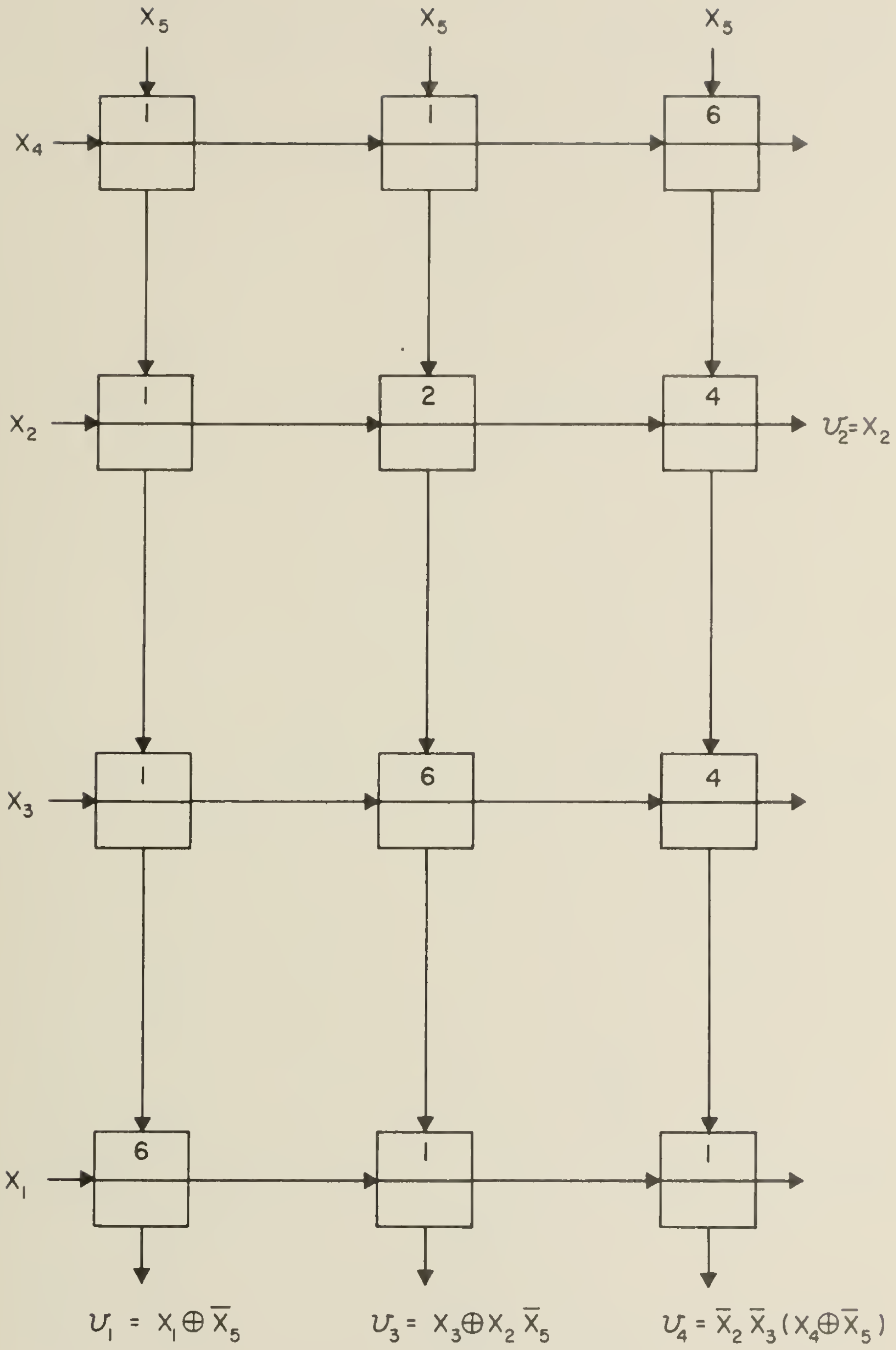


FIGURE 24. HEXAGONAL NINES COMPLEMENT CIRCUIT.



X_5
X_5
X_5
X_4
1
1
6
X_2
1
2
4
$U_2 = X_2$
X_3
1
6
4
X_1
6
1
1
$U_1 = X_1 \oplus \overline{X}_5$
$U_3 = X_3 \oplus X_2 \overline{X}_5$
$U_4 = \overline{X}_2 \overline{X}_3 (X_4 \oplus \overline{X}_5)$

FIGURE 25. CUTPOINT CELL NINES COMPLEMENT CIRCUIT *

* Minnick, R.C., "Cutpoint Cellular Logic," p.695.

$W_1 = \bar{x}_3\left[\bar{x}_1 + (\bar{x}_2 \oplus x_4)\right] \quad W_2 = x_1(\bar{x}_2 + \bar{x}_3) \quad W_3 = x_2 \quad W_4 = x_3(\bar{x}_1 + \bar{x}_2) + \bar{x}_1\bar{x}_2\bar{x}_4 \quad W_5 = x_4 + x_3(x_1 \oplus \bar{x}_2 \oplus x_4)$

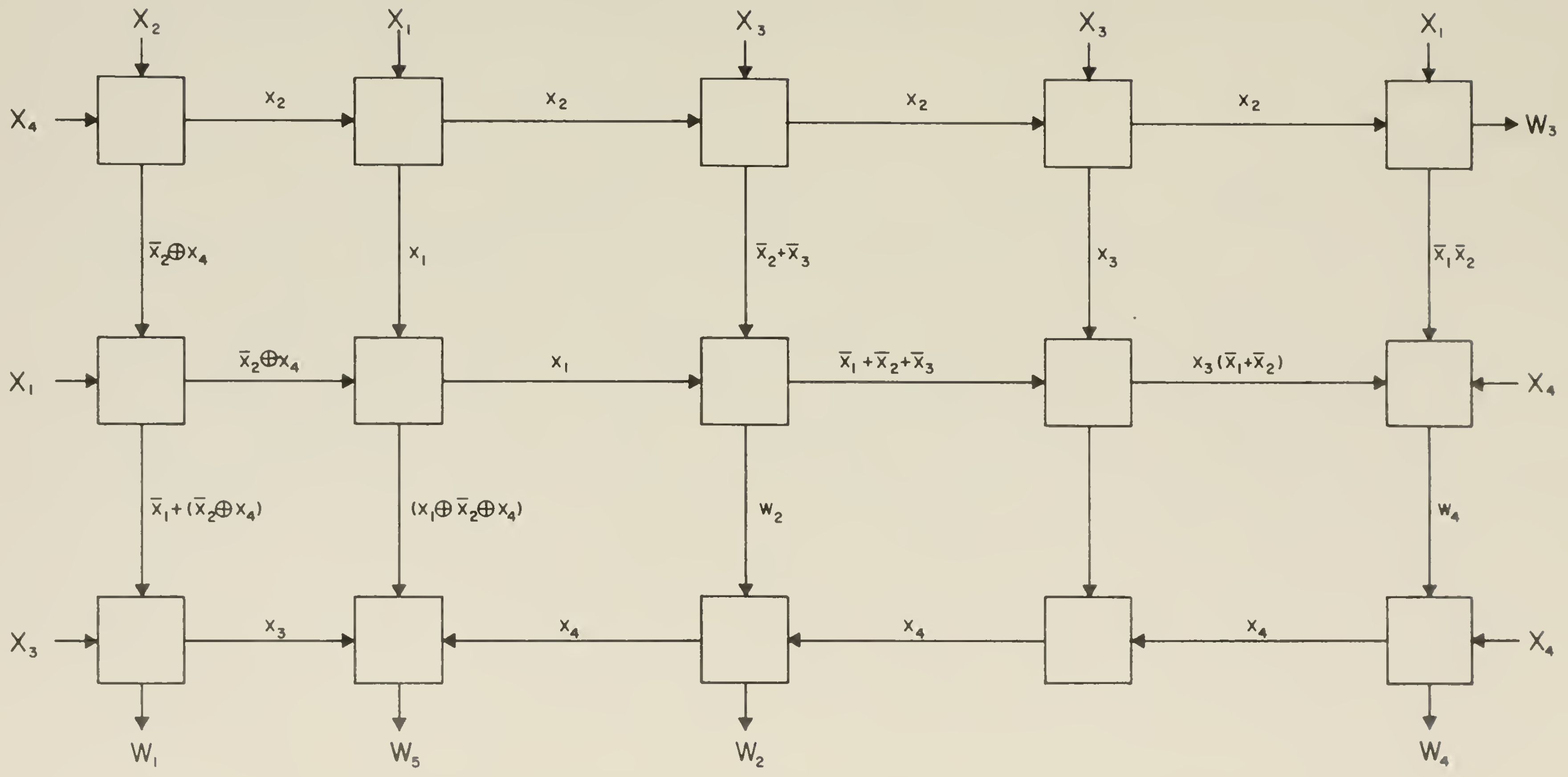


FIGURE 26. SQUARE CELL TRANSLATOR FROM BINARY-CODED DECIMAL TO TWO-OUT-OF-FIVE.

$W_1 = \bar{X}_3[\bar{X}_1 + (\bar{X}_2 \oplus X_4)]$ $\quad W_2 = X_1(\bar{X}_2 + \bar{X}_3)$ $\quad W_3 = X_2$ $\quad W_4 = X_3(\bar{X}_1 + \bar{X}_2) + \bar{X}_1\bar{X}_2\bar{X}_4$ $\quad W_5 = X_4 + X_3(X_1 \oplus \bar{X}_2 \oplus X_4)$

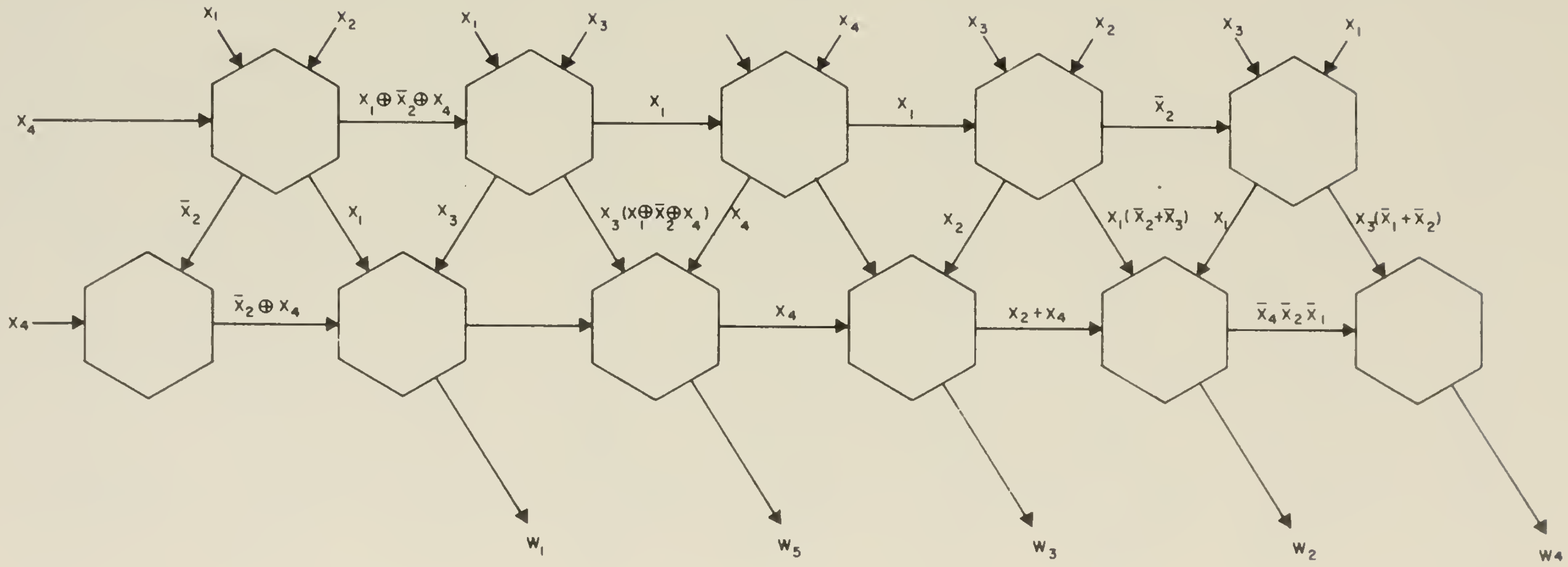


FIGURE 27. HEXAGONAL CELL TRANSLATOR FROM BINARY-CODED DECIMAL TO TWO-OUT-OF-FIVE.

$W_1 = \bar{x}_3[\bar{x}_1 + (\bar{x}_2 \oplus x_4)]$ $W_2 = x_1(\bar{x}_2 + \bar{x}_3)$ $W_3 = x_2$ $W_4 = x_3(\bar{x}_1 + \bar{x}_2) + \bar{x}_1\bar{x}_2\bar{x}_4$ $W_5 = x_4 + x_3(x_1 \oplus \bar{x}_2 \oplus x_4)$

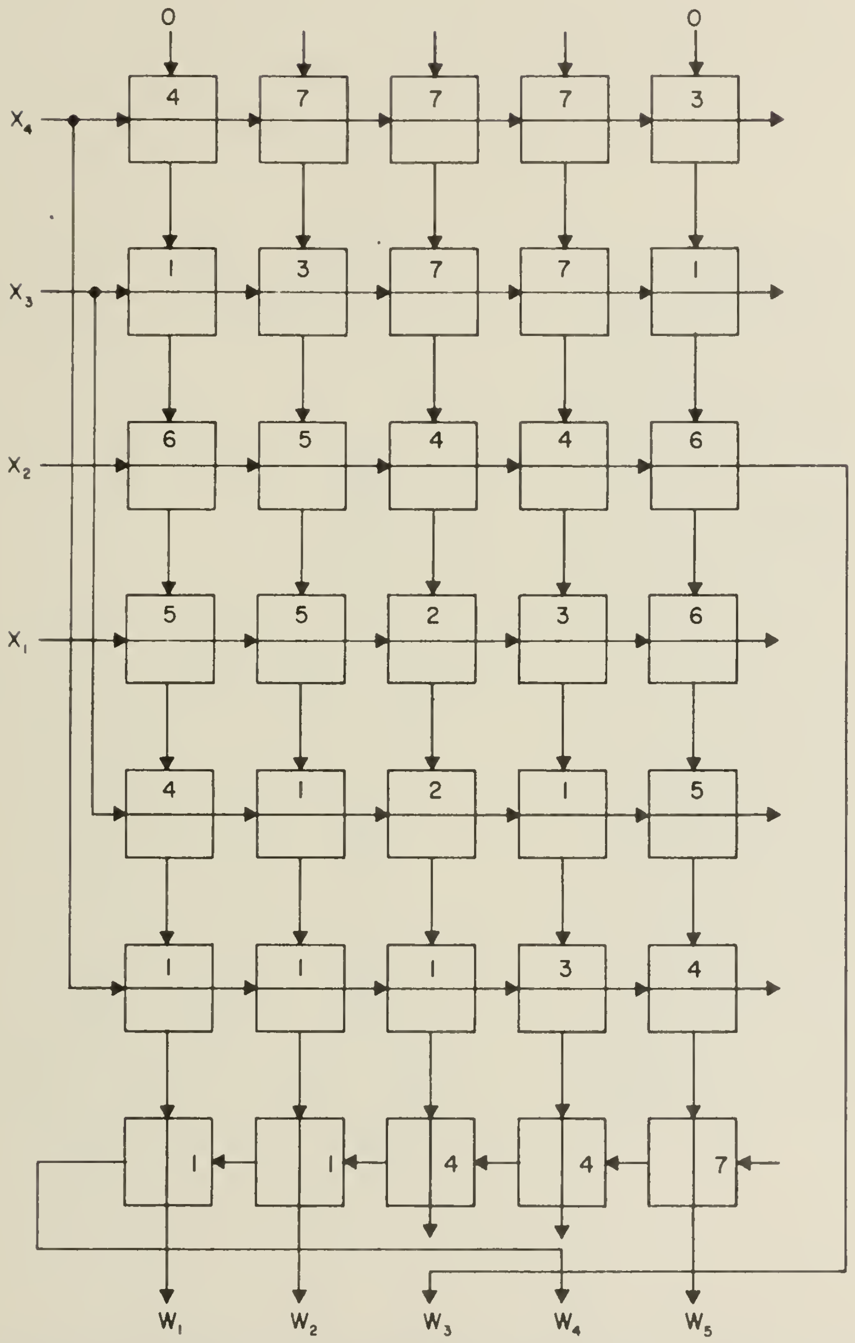
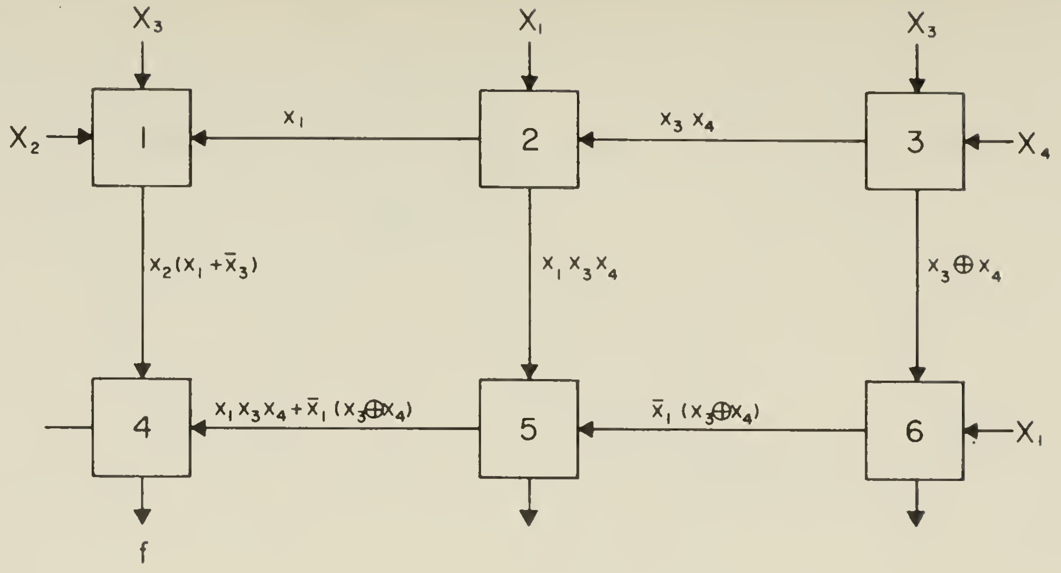


FIGURE 28. CUTPOINT CELL TRANSLATOR FROM BINARY-CODED DECIMAL TO TWO-OUT-OF-FIVE *

* Minnick, R.C., "Cutpoint Cellular Logic," p.696

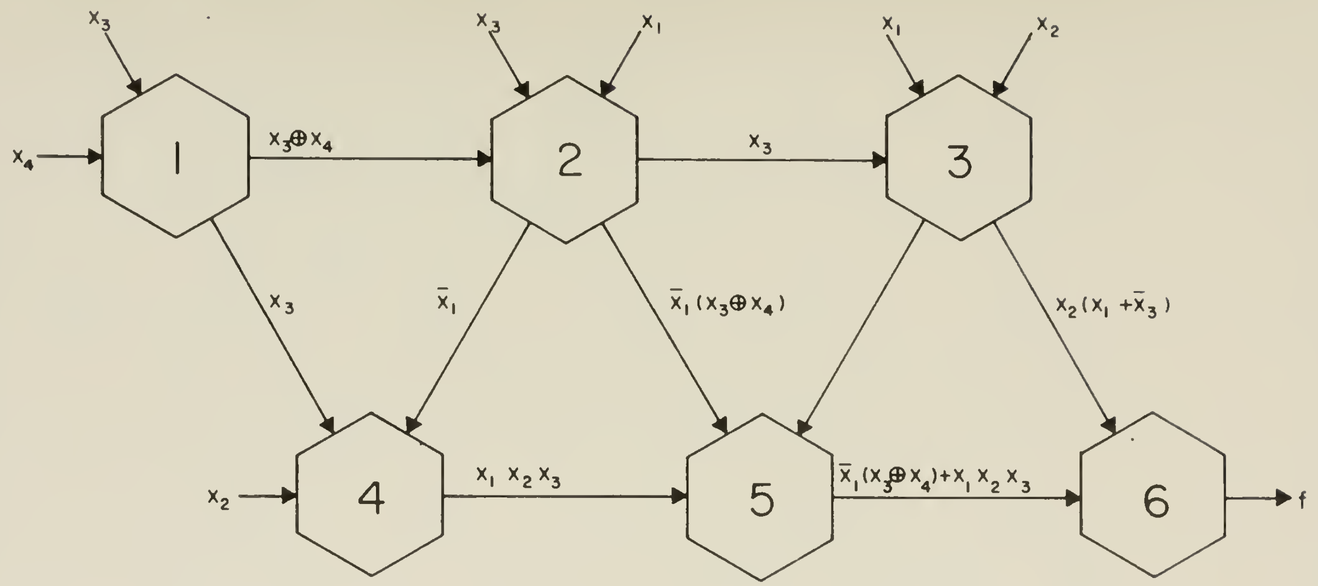
$f = \bar{x}_1 \bar{x}_3 x_4 + \bar{x}_1 x_3 \bar{x}_4 + x_1 x_3 x_4 + x_1 x_2 + x_2 \bar{x}_3 = x_2(x_1 + \bar{x}_3) + \bar{x}_1(x_3 \oplus x_4) + x_1 x_3 x_4$



CODE / BLOCK	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	0	1	1	d	d	1	1	1	0	0	0	0	0	0	1	1
2	1	1	1	d	d	0	1	1	1	1	1	0	0	1	1	1
3	1	1	1	d	d	0	0	1	1	0	1	0	0	1	1	1
4	1	1	1	d	d	0	d	d	d	1	0	0	0	1	1	1
5	1	1	1	d	d	0	0	1	0	d	d	d	0	1	1	1
6	1	1	0	d	d	0	0	1	1	d	d	d	0	1	1	1

FIGURE 29. SQUARE CELL EXAMPLE A

$f = \bar{x}_1 \bar{x}_3 x_4 + \bar{x}_1 x_3 \bar{x}_4 + x_1 x_3 x_4 + x_1 x_2 + x_2 \bar{x}_3 = x_2(x_1 + \bar{x}_3) + \bar{x}_1(x_3 \oplus x_4) + x_1 x_2 x_3$



CODE / CELL	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1	1	d	1	1	0	0	1	1	0	1	0	0	1	0	0
2	1	1	0	0	1	0	0	1	0	1	0	1	0	0	1	0
3	0	1	1	d	d	d	d	d	d	1	1	1	1	0	1	0
4	1	1	0	1	1	1	0	0	1	d	d	d	d	d	d	0
5	1	1	d	1	1	0	0	1	0	d	d	d	d	d	d	0
6	1	1	d	1	1	0	0	1	0	d	d	d	d	d	d	0

FIGURE 30. HEXAGONAL CELL EXAMPLE A

$$f = \bar{x}_1 \bar{x}_3 x_4 + \bar{x}_1 x_3 \bar{x}_4 + x_1 x_3 x_4 + x_1 x_2 + x_2 \bar{x}_3 = x_4(\overline{x_1 \oplus x_3}) + \bar{x}_1 x_3 \bar{x}_4 + x_1 x_2 + x_2 \bar{x}_3$$

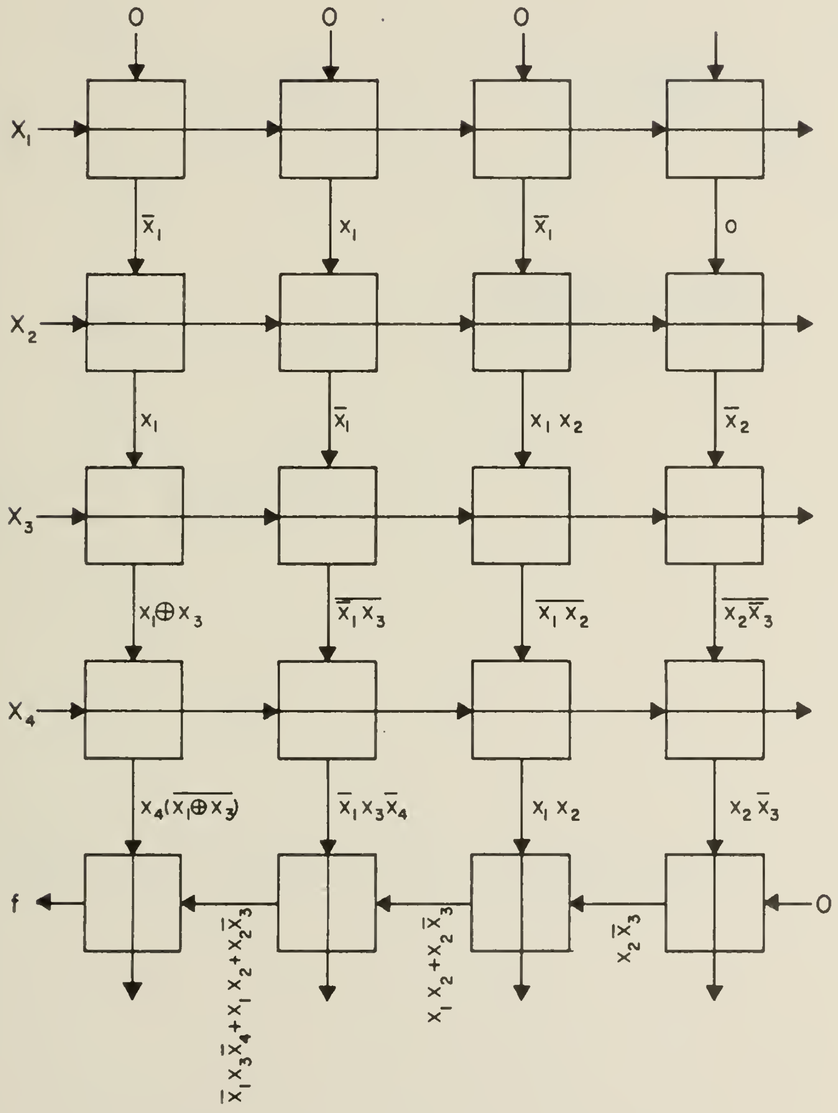

0
0
0
X_1
$\bar{x}_1$
x_1
$\bar{x}_1$
0
X_2
x_1
$\bar{x}_1$
$x_1 x_2$
$\bar{x}_2$
X_3
$x_1 \oplus x_3$
$\overline{\bar{x}_1 x_3}$
$\overline{x_1 x_2}$
$\overline{x_2 \bar{x}_3}$
X_4
$x_4(\overline{x_1 \oplus x_3})$
$\bar{x}_1 x_3 \bar{x}_4$
$x_1 x_2$
$x_2 \bar{x}_3$
f
0
$\bar{x}_1 x_3 \bar{x}_4 + x_1 x_2 + x_2 \bar{x}_3$
$x_1 x_2 + x_2 \bar{x}_3$
$x_2 \bar{x}_3$

FIGURE 31. CUTPOINT CELL EXAMPLE A

$$f = X_1 X_2 X_3 + X_1 \bar{X}_2 X_3 X_4 + X_1 X_2 \bar{X}_3 \bar{X}_4 + \bar{X}_1 \bar{X}_2 \bar{X}_3 \bar{X}_4 = X_1 X_3 X_4 + (\overline{X_1 + X_2})(\overline{X_3 \oplus X_4})$$

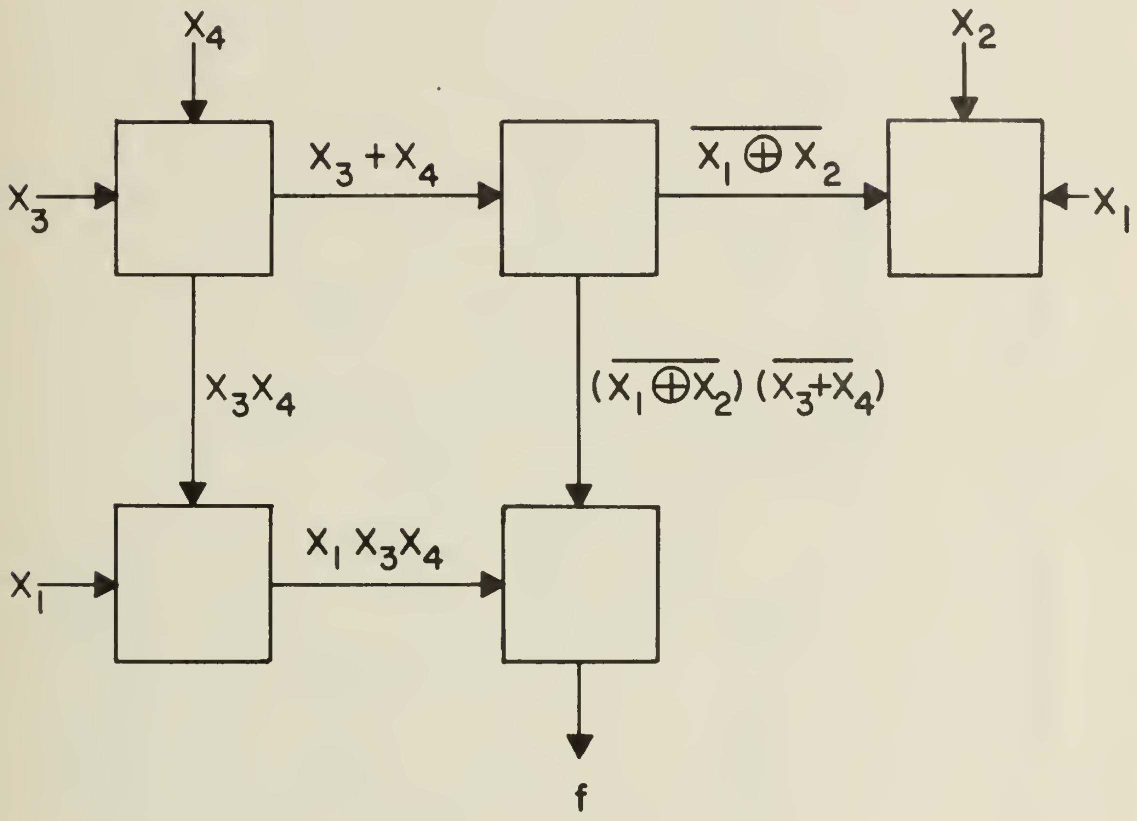

X_4
X_2
X_3
$X_3 + X_4$
$\overline{X_1 \oplus X_2}$
X_1
$X_3 X_4$
$(\overline{X_1 \oplus X_2})(\overline{X_3 + X_4})$
X_1
$X_1 X_3 X_4$
f

FIGURE 32. SQUARE CELL EXAMPLE B

$$f = X_1 X_3 X_4 + \overline{(X_1 \oplus X_2)}\,\overline{(X_3 + X_4)}$$

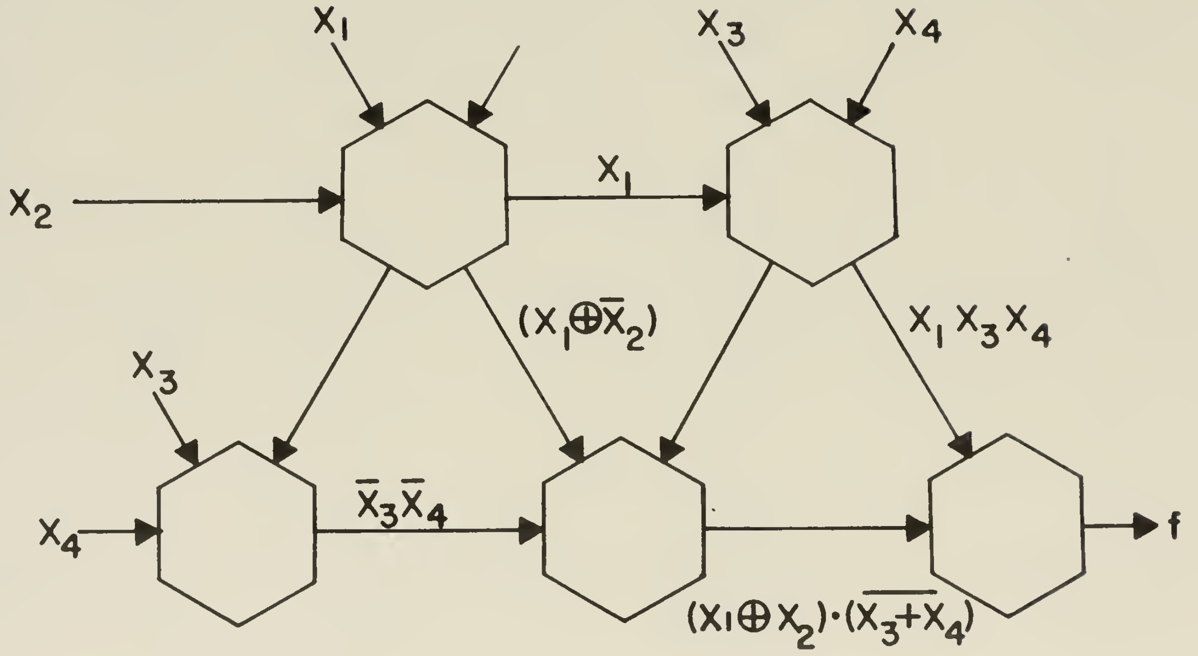
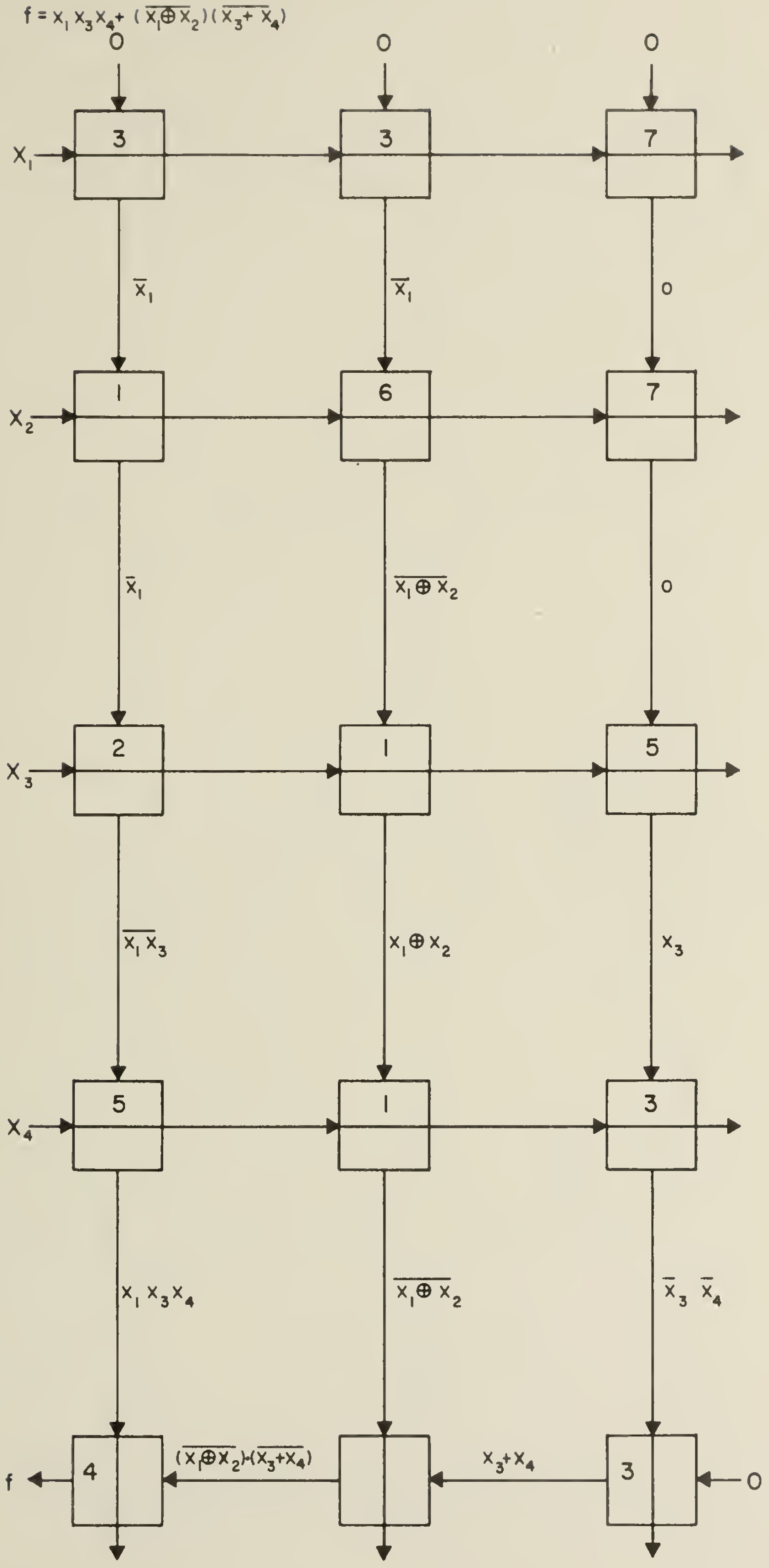


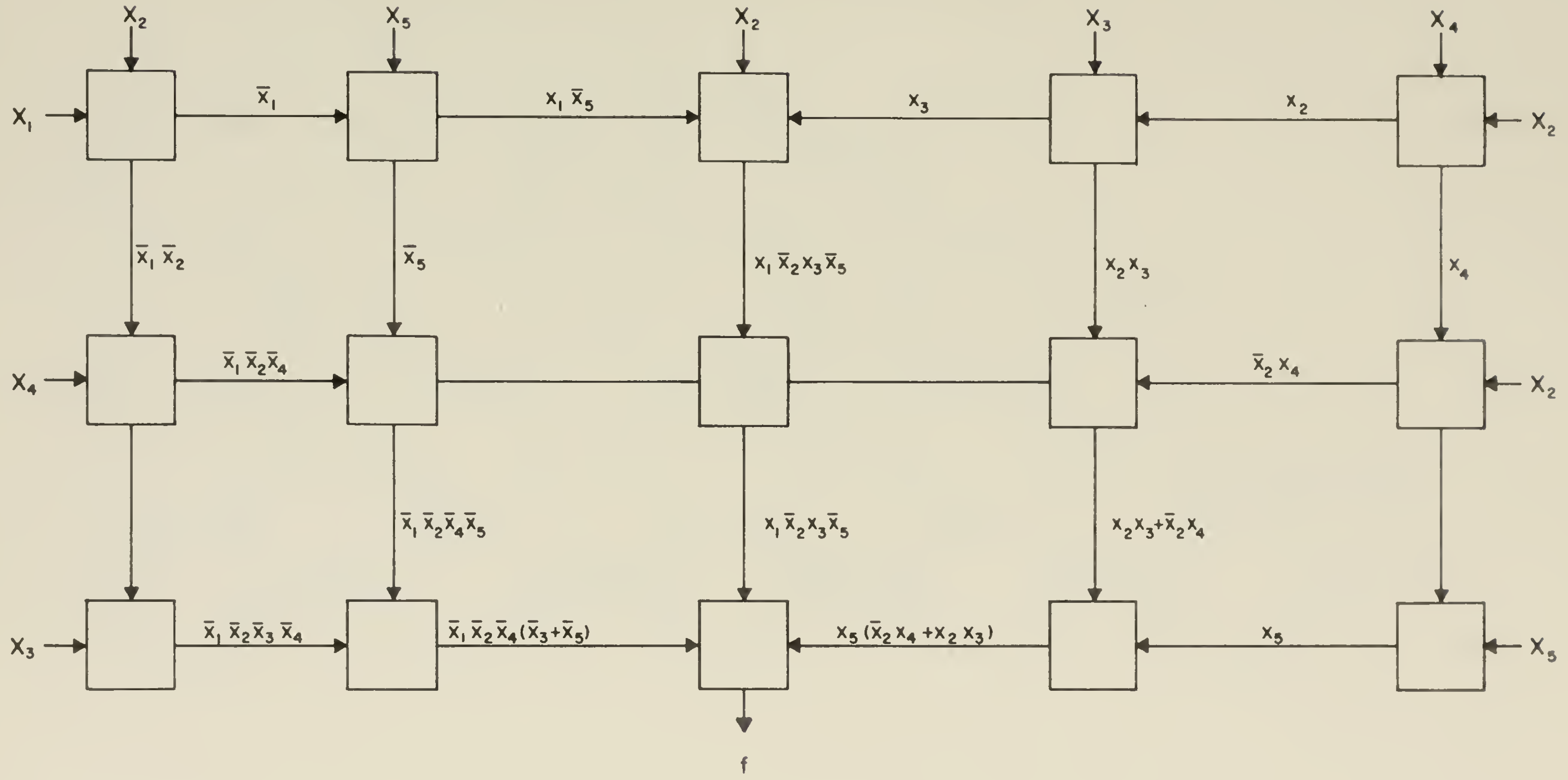
FIGURE 33. HEXAGONAL CELL EXAMPLE B



$f = x_1 x_3 x_4 + (\overline{x_1 \oplus x_2})(\overline{x_3 + x_4})$
0
0
0
X_1
3
3
7
$\bar{x}_1$
$\bar{x}_1$
0
X_2
1
6
7
$\bar{x}_1$
$\overline{x_1 \oplus x_2}$
0
X_3
2
1
5
$\overline{x_1 x_3}$
$x_1 \oplus x_2$
x_3
X_4
5
1
3
$x_1 x_3 x_4$
$\overline{x_1 \oplus x_2}$
$\bar{x}_3 \ \bar{x}_4$
f
4
$(\overline{x_1 \oplus x_2}) \cdot (\overline{x_3 + x_4})$
$x_3 + x_4$
3
0

FIGURE. 34 CUTPOINT CELL EXAMPLE B

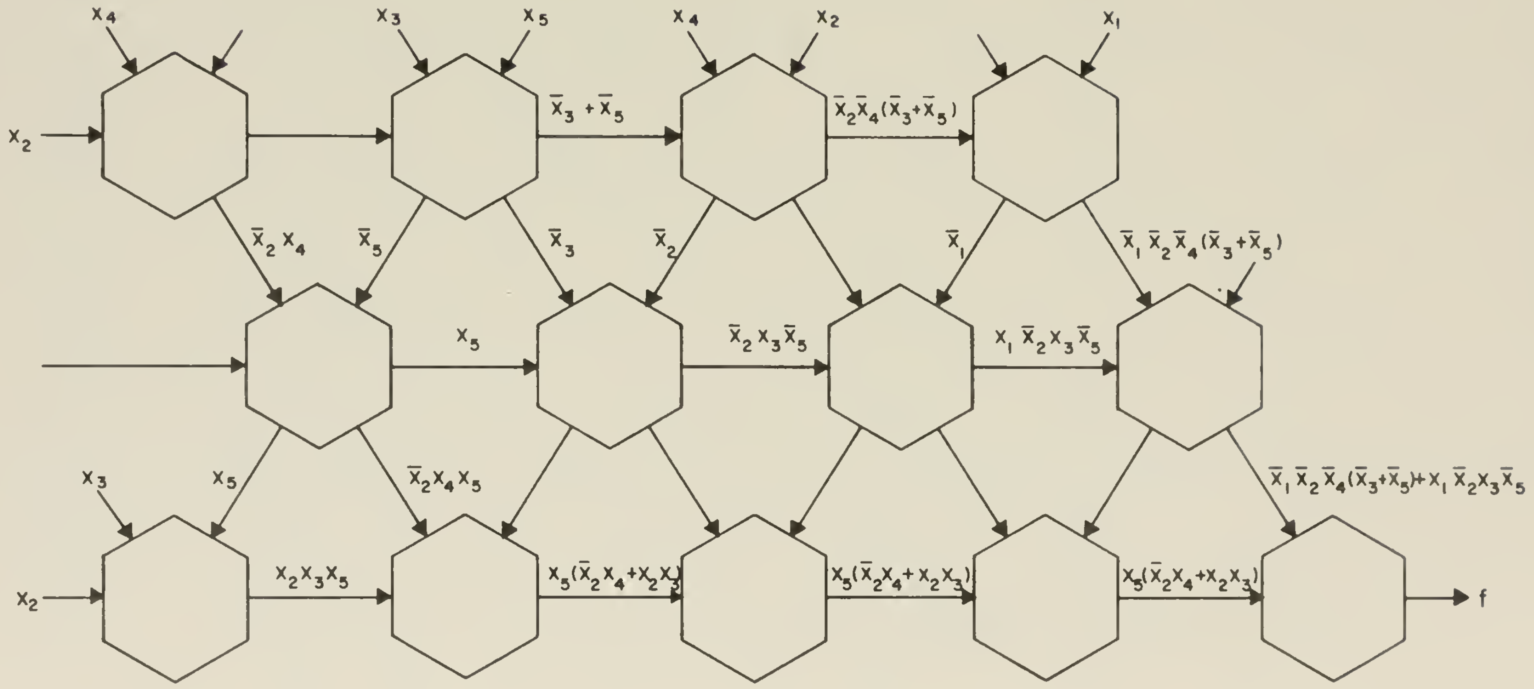
$$f=\bar{x}_2x_4x_5+x_2x_3x_5+\bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4+\bar{x}_1\bar{x}_2\bar{x}_4\bar{x}_5+x_1\bar{x}_2x_3\bar{x}_5=x_5(\bar{x}_2x_4+x_2x_3)+\bar{x}_1\bar{x}_2\bar{x}_4(\bar{x}_3+\bar{x}_5)+x_1\bar{x}_2+x_3\bar{x}_5$$



X_2
X_5
X_2
X_3
X_4
X_1
$\bar{x}_1$
$x_1\bar{x}_5$
x_3
x_2
X_2
$\bar{x}_1\bar{x}_2$
$\bar{x}_5$
$x_1\bar{x}_2x_3\bar{x}_5$
x_2x_3
x_4
X_4
$\bar{x}_1\bar{x}_2\bar{x}_4$
$\bar{x}_2x_4$
X_2
$\bar{x}_1\bar{x}_2\bar{x}_4\bar{x}_5$
$x_1\bar{x}_2x_3\bar{x}_5$
$x_2x_3+\bar{x}_2x_4$
X_3
$\bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4$
$\bar{x}_1\bar{x}_2\bar{x}_4(\bar{x}_3+\bar{x}_5)$
$x_5(\bar{x}_2x_4+x_2x_3)$
x_5
X_5
f

FIGURE 35. SQUARE CELL EXAMPLE C

$$f=\bar{x}_2 x_4 x_5+x_2 x_3 x_5+\bar{x}_1 \bar{x}_2 \bar{x}_3 \bar{x}_4+\bar{x}_1 \bar{x}_2 \bar{x}_4 \bar{x}_5+x_1 \bar{x}_2 x_3 \bar{x}_5=x_5(\bar{x}_2 x_4+x_2 x_3)+\bar{x}_1 \bar{x}_2 \bar{x}_4(\bar{x}_3+\bar{x}_5)+x_1 \bar{x}_2 x_3 \bar{x}_5$$



x_4
x_3
x_5
x_4
x_2
x_1
x_2
$\bar{x}_3+\bar{x}_5$
$\bar{x}_2\bar{x}_4(\bar{x}_3+\bar{x}_5)$
$\bar{x}_2 x_4$
$\bar{x}_5$
$\bar{x}_3$
$\bar{x}_2$
$\bar{x}_1$
$\bar{x}_1\bar{x}_2\bar{x}_4(\bar{x}_3+\bar{x}_5)$
x_5
$\bar{x}_2 x_3 \bar{x}_5$
$x_1 \bar{x}_2 x_3 \bar{x}_5$
x_3
x_5
$\bar{x}_2 x_4 x_5$
$\bar{x}_1\bar{x}_2\bar{x}_4(\bar{x}_3+\bar{x}_5)+x_1\bar{x}_2 x_3\bar{x}_5$
x_2
$x_2 x_3 x_5$
$x_5(\bar{x}_2 x_4+x_2 x_3)$
$x_5(\bar{x}_2 x_4+x_2 x_3)$
$x_5(\bar{x}_2 x_4+x_2 x_3)$
f

FIGURE 36. HEXAGONAL CELL EXAMPLE C

$f = \bar{x}_2 x_4 x_5 + x_2 x_3 x_5 + \bar{x}_1 \bar{x}_2 \bar{x}_3 \bar{x}_4 + \bar{x}_1 \bar{x}_2 \bar{x}_4 \bar{x}_5 + x_1 \bar{x}_2 x_3 \bar{x}_5$

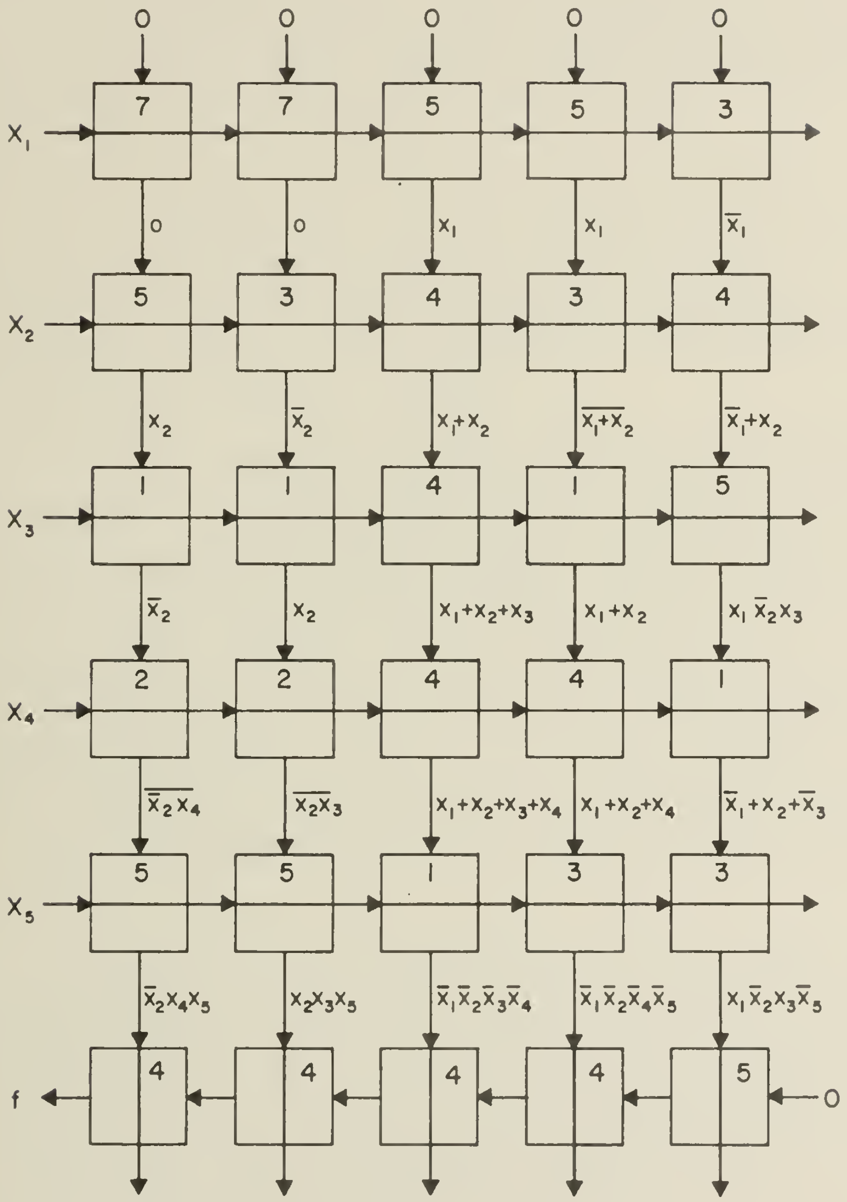

0 0 0 0 0
X1
7 7 5 5 3
0 0 x1 x1 x̄1
X2
5 3 4 3 4
x2 x̄2 x1+x2 (x1+x2)‾ x̄1+x2
X3
1 1 4 1 5
x̄2 x2 x1+x2+x3 x1+x2 x1x̄2x3
X4
2 2 4 4 1
(x2x4)‾ (x2x3)‾ x1+x2+x3+x4 x1+x2+x4 x̄1+x2+x̄3
X5
5 5 1 3 3
x̄2x4x5 x2x3x5 x̄1x̄2x̄3x̄4 x̄1x̄2x̄4x̄5 x1x̄2x3x̄5
f
4 4 4 4 5
0

FIGURE 37. CUTPOINT CELL EXAMPLE C

6. SUMMARY AND CONCLUSION

Three types of cells, cutpoint , hexagonal, and square, were discussed in this thesis. Each of these designs were proposed to replace the large number of individual integrated chips that are in use today.

The cutpoint cell is the most primitive cell. Only half of the functions of two variables and a flip-flop function are obtainable from this cell. This results in cellular arrays for a n-variable function as large as n+1 by 2^{n-2} cells. At present, though, this is the easiest module to program.

The square cell is the next step in refining this logical design technique. One fixed input, one fixed output, and two variable lines are employed in producing many of the functions of three variables. As a result, 56% less square cells than cutpoint cells are used in typical examples. Since no algorithm exists at present, programming these cells must be done by hand. Research into these programming requirements, however, may uncover a suitable computer algorithm. Also, new minimization techniques may reduce the number of cells needed to produce various functions and thus make programming easier.

The final step in the production of a new logic element is the hexagonal cell. Although this cell produces less functions than the square cell, its three fixed inputs and three fixed outputs yield arrays with 66% less cells than cutpoint arrays and 10% less cells than square cell arrays. Once again, an algorithm for a computer is needed to eliminate programming by hand.

A further improvement can be made in both the square and hexagonal cells by increasing the logic in each cell. At present around 200 gates are used in the design of the logic. Since this large number of logic elements appears to be the maximum that can be put on a chip at present, perhaps future innovation could increase this number. With the increased number of gates, however, goes an increase in the control bits, tending to complicate the program and preventing any substantial saving. The 16 bits used in the hexagonal and square cells were the minimum number of bits that would yield the necessary basic functions.

Although an internal shift register is included in each cell, it is by no means the only possible method. In fact, it might not even be the best way as future research may prove. Other suggestions for cell addressing would include external registers, microwelding, or even 16 individual lines per cell.

Finally, it is possible that shapes other than square or hexagonal would be more useful. Another area for investigation would be multilayer designs. In this case, the cells that comprise one layer may even be different in shape from those of the adjoining ones.

Thus, in this paper it has been shown that a single chip capable of performing various functions can be used as the basis for digital circuits. These cells would eliminate the large number of individual NAND and NOR gates and inverters that are presently encountered in this type of circuitry. As a result, the digital equipment could be smaller in size. Also, after an initial period, the cost per module would approach that of the individual integrated chips and would yield a substantial monetary saving. As a final comment on the advisability of using modular design, Mr. John Holland states:

"If the cost of production is largely set-up cost, it may be possible to produce complicated modules for what it presently costs to produce and assemble a few transistors. Should this happen, average use factor for individual elements is no longer a reasonable measure of overall machine efficiency."*

*Holland, John H., "Iterative Circuit Computers: Characterization and Resume of Advantages and Disadvantages", in Spandorfer, L. M., _Proc. of a Symposium on Microelectronics and Large Systems_, p. 176.

REFERENCES

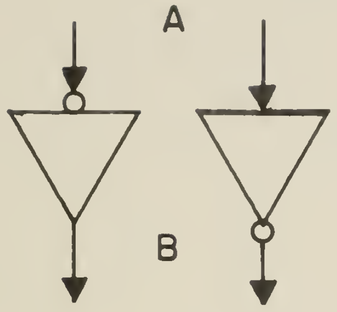
1. Minnick, R.C., "Cutpoint Cellular Logic", IEEE Transactions on Electron Computers, EC-13, Vol. 6, pp. 685-698, December 1964.

2. The Staff of the Computation Laboratory, "Synthesis of Electronic Computing and Control Circuits", Harvard University Press, Cambridge, Mass., 1951, pp. 23-27.

3. Holland, John H., "Iterative Circuit Computers: Characterization and Resume of Advantages and Disadvantages", in Spandorfer, L. M., Proc. of a Symposium on Microelectronics and Large Systems, Spartan Books, Washington, D.C., 1965, pp. 175-177.

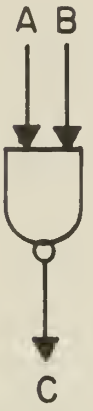
4. Hohn, Franz E., "Applied Boolean Algebra", Second Edition, The Macmillan Company, New York, 1966.

APPENDIX A
LOGIC ELEMENTS



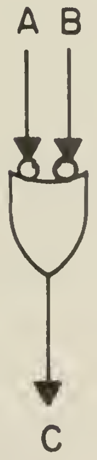
INVERTER

A = +4V,	B = 0V
A = 0V,	B = +4V



NAND GATE

A = 0V,	B = 0V,	C = +4V
A = 0V,	B = +4V,	C = +4V
A = +4V,	B = 0V,	C = +4V
A = +4V,	B = +4V,	C = 0V



NOR GATE

A = 0V,	B = 0V,	C = +4V
A = 0V,	B = +4V,	C = +4V
A = +4V,	B = 0V,	C = +4V
A = +4V,	B = +4V,	C = 0V

APPENDIX B

Number of Cells Used to Form a Function of Three Variables

Class	Hexagonal Cell	Square Cell
0	1	1
1	1	1
2	1	1
3	2	1
4	1	1
5	1	1
6	2	2
7	1	1
8	1	1
9	2	2
10	2	2
11	2	1
12	1	1
13	1	1
14	2	1
15	2	2
16	2	2
17	1	1
18	2	2
19	2	2
20	1	1
21	1	1

APPENDIX B Cont'd

Summary of the 256 Functions of Three Variables

Class	Number of Functions	Representative Function
0	1	0
1	8	xyz
2	4	$xyz + \bar{x}\bar{y}\bar{z}$
3	12	$x(y \oplus z)$
4	12	xy
5	8	$x(y \oplus z) + \bar{x}yz$
6	24	$xy + z\bar{x}\bar{y}$
7	24	$x(y + z)$
8	6	x
9	8	$yz + x(y + z)$
10	24	$xy + \bar{x}z$
11	24	$x \oplus yz$
12	2	$x \oplus (y \oplus z)$
13	6	$x \oplus y$
14	24	$x + yz$
15	24	$xy + (y \oplus z)$
16	8	$yz + (x \oplus yz)$
17	12	$x + y$
18	12	$x + (y \oplus z)$
19	4	$(x \oplus y) + (x \oplus z)$
20	8	$x + y + z$
21	1	1

APPENDIX B Cont'd

Functions of Three Variables

Class 0:	0	
Class 1:	xyz	$x\bar{y}z$
	$xy\bar{z}$	
	$x\bar{y}\bar{z}$	
	$\bar{x}yz$	
	$\bar{x}y\bar{z}$	
	$\bar{x}\bar{y}z$	
	$\bar{x}\bar{y}\bar{z}$	
Class 2:	$xyz + \bar{x}\bar{y}\bar{z}$	
	$\bar{x}yz + x\bar{y}\bar{z}$	
	$x\bar{y}z + \bar{x}y\bar{z}$	
	$xy\bar{z} + \bar{x}\bar{y}z$	
Class 3:	$x(y\oplus z)$	$z(\overline{x\oplus y})$
	$x(\overline{y\oplus z})$	$\bar{z}(x\oplus y)$
	$\bar{x}(y\oplus z)$	$\bar{z}(\overline{x\oplus y})$
	$\bar{x}(\overline{y\oplus z})$	
	$y(x\oplus z)$	
	$y(\overline{x\oplus z})$	
	$\bar{y}(x\oplus z)$	
	$\bar{y}(\overline{x\oplus z})$	
	$z(x\oplus y)$	

Class 4:	xy
	$x\bar{y}$
	$\bar{x}y$
	$\bar{x}\bar{y}$
	yz
	$y\bar{z}$
	$\bar{y}z$
	$\bar{y}\bar{z}$
	xz
	$x\bar{z}$
	$\bar{x}z$
	$\bar{x}\bar{z}$
Class 5:	$x(y\oplus z) + \bar{x}yz$
	$x(y\oplus z) + \bar{x}\bar{y}z$
	$x(y\oplus z) + \bar{x}y\bar{z}$
	$x(y\oplus z) + \bar{x}\bar{y}\bar{z}$
	$x(\overline{y\oplus z}) + \bar{x}yz$
	$x(\overline{y\oplus z}) + \bar{x}\bar{y}z$
	$x(\overline{y\oplus z}) + \bar{x}y\bar{z}$
	$x(\overline{y\oplus z}) + \bar{x}\bar{y}\bar{z}$

Class 6:

$xy + z\bar{x}\bar{y}$

$xy + \bar{z}\bar{x}\bar{y}$

$x\bar{y} + z\bar{x}y$

$x\bar{y} + \bar{z}\bar{x}y$

$\bar{x}y + zx\bar{y}$

$\bar{x}y + \bar{z}x\bar{y}$

$\bar{x}\bar{y} + zxy$

$\bar{x}\bar{y} + \bar{z}xy$

$yz + x\bar{y}\bar{z}$

$yz + \bar{x}\bar{y}\bar{z}$

$y\bar{z} + x\bar{y}z$

$y\bar{z} + \bar{x}\bar{y}z$

$\bar{y}z + xy\bar{z}$

$\bar{y}z + \bar{x}y\bar{z}$

$\bar{y}\bar{z} + xyz$

$\bar{y}\bar{z} + \bar{x}yz$

$xz + y\bar{x}\bar{z}$

$xz + \bar{y}\bar{x}\bar{z}$

$x\bar{z} + y\bar{x}z$

$x\bar{z} + \bar{y}\bar{x}z$

$\bar{x}z + yx\bar{z}$

$\bar{x}z + \bar{y}x\bar{z}$

$\bar{x}\bar{z} + yxz$

$\bar{x}\bar{z} + \bar{y}xz$

Class 7:

$x(y+z)$	$\bar{x}(\bar{y}+\bar{z})$	$\bar{y}(x+\bar{z})$	$\bar{z}(\bar{x}+y)$
$x(\bar{y}+z)$	$y(x+z)$	$\bar{y}(\bar{x}+\bar{z})$	$\bar{z}(x+\bar{y})$
$x(y+\bar{z})$	$y(\bar{x}+z)$	$z(x+y)$	$\bar{z}(\bar{x}+\bar{y})$
$x(\bar{y}+\bar{z})$	$y(x+\bar{z})$	$z(\bar{x}+y)$	
$\bar{x}(y+z)$	$y(\bar{x}+\bar{z})$	$z(x+\bar{y})$	
$\bar{x}(\bar{y}+z)$	$\bar{y}(x+z)$	$z(\bar{x}+\bar{y})$	
$\bar{x}(y+\bar{z})$	$\bar{y}(\bar{x}+z)$	$\bar{z}(x+y)$	

Class 8:	x	
	$\bar{x}$	
	y	
	$\bar{y}$	
	z	
	$\bar{z}$	
Class 9:	$x(y+z)+yz$	
	$x(\bar{y}+z)+\bar{y}z$	
	$x(y+\bar{z})+y\bar{z}$	
	$x(\bar{y}+\bar{z})+\bar{y}\bar{z}$	
	$\bar{x}(y+z)+yz$	
	$\bar{x}(\bar{y}+z)+\bar{y}z$	
	$\bar{x}(y+\bar{z})+y\bar{z}$	
	$\bar{x}(\bar{y}+\bar{z})+\bar{y}\bar{z}$	
Class 10:	$xy + \bar{x}z$	$zx + \bar{z}y$
	$xy + \bar{x}\bar{z}$	$zx + \bar{z}\bar{y}$
	$x\bar{y} + \bar{x}z$	$z\bar{x} + \bar{z}y$
	$x\bar{y} + \bar{x}\bar{z}$	$z\bar{x} + \bar{z}\bar{y}$
	$\bar{x}y + xz$	$\bar{z}x + zy$
	$\bar{x}y + x\bar{z}$	$\bar{z}x + z\bar{y}$
	$\bar{x}\bar{y} + xz$	$\bar{z}\bar{x} + zy$
	$\bar{x}\bar{y} + x\bar{z}$	$\bar{z}\bar{x} + z\bar{y}$
	$yz + \bar{y}x$	
	$yz + \bar{y}\bar{x}$	
	$y\bar{z} + \bar{y}x$	
	$y\bar{z} + \bar{y}\bar{x}$	
	$\bar{y}z + yx$	
	$\bar{y}z + y\bar{x}$	
	$\bar{y}\bar{z} + yx$	
	$\bar{y}\bar{z} + y\bar{x}$	

Class 11:	$x \oplus yz$	$z \oplus xy$
	$x \oplus y\bar{z}$	$z \oplus x\bar{y}$
	$x \oplus \bar{y}z$	$z \oplus \bar{x}y$
	$x \oplus \bar{y}\bar{z}$	$z \oplus \bar{x}\bar{y}$
	$\bar{x} \oplus yz$	$\bar{z} \oplus xy$
	$\bar{x} \oplus y\bar{z}$	$\bar{z} \oplus x\bar{y}$
	$\bar{x} \oplus \bar{y}z$	$\bar{z} \oplus \bar{x}y$
	$\bar{x} \oplus \bar{y}\bar{z}$	$\bar{z} \oplus \bar{x}\bar{y}$
	$y \oplus xz$	
	$y \oplus x\bar{z}$	
	$y \oplus \bar{x}z$	
	$y \oplus \bar{x}\bar{z}$	
	$\bar{y} \oplus xz$	
	$\bar{y} \oplus x\bar{z}$	
	$\bar{y} \oplus \bar{x}z$	
	$\bar{y} \oplus \bar{x}\bar{z}$	
Class 12:	$x \oplus (y \oplus z)$	
	$x \oplus (\overline{y \oplus z})$	
Class 13:	$x \oplus y$	
	$\overline{x \oplus y}$	
	$y \oplus z$	
	$\overline{y \oplus z}$	
	$x \oplus z$	
	$\overline{x \oplus z}$	

Class 14:	$x + yz$	$\bar{y} + xz$
	$x + y\bar{z}$	$\bar{y} + x\bar{z}$
	$x + \bar{y}z$	$\bar{y} + \bar{x}z$
	$x + \bar{y}\bar{z}$	$\bar{y} + \bar{x}\bar{z}$
	$\bar{x} + yz$	$z + xy$
	$\bar{x} + y\bar{z}$	$z + x\bar{y}$
	$\bar{x} + \bar{y}z$	$z + \bar{x}y$
	$\bar{x} + \bar{y}\bar{z}$	$z + \bar{x}\bar{y}$
	$y + xz$	$\bar{z} + xy$
	$y + x\bar{z}$	$\bar{z} + x\bar{y}$
	$y + \bar{x}z$	$\bar{z} + \bar{x}y$
	$y + \bar{x}\bar{z}$	$\bar{z} + \bar{x}\bar{y}$
Class 15:	$xy + (y\oplus z)$	$x\bar{z} + (x\oplus y)$
	$xy + (\overline{y\oplus z})$	$x\bar{z} + (\overline{x\oplus y})$
	$x\bar{y} + (y\oplus z)$	$\bar{x}z + (x\oplus y)$
	$x\bar{y} + (\overline{y\oplus z})$	$\bar{x}z + (\overline{x\oplus y})$
	$\bar{x}y + (y\oplus z)$	$\bar{x}\bar{z} + (x\oplus y)$
	$\bar{x}y + (\overline{y\oplus z})$	$\bar{x}\bar{z} + (\overline{x\oplus y})$
	$\bar{x}\bar{y} + (y\oplus z)$	
	$\bar{x}\bar{y} + (\overline{y\oplus z})$	
	$yz + (x\oplus z)$	
	$yz + (\overline{x\oplus z})$	
	$y\bar{z} + (x\oplus z)$	
	$y\bar{z} + (\overline{x\oplus z})$	
	$\bar{y}z + (x\oplus z)$	
	$\bar{y}z + (\overline{x\oplus z})$	
	$\bar{y}\bar{z} + (x\oplus z)$	
	$\bar{y}\bar{z} + (\overline{x\oplus z})$	
	$xz + (x\oplus y)$	
	$xz + (\overline{x\oplus y})$	

Class 16:

$yz + (x \oplus yz)$

$yz + (\overline{x \oplus yz})$

$y\bar{z} + (x \oplus yz)$

$y\bar{z} + (\overline{x \oplus yz})$

$\bar{y}z + (x \oplus yz)$

$\bar{y}z + (\overline{x \oplus yz})$

$\bar{y}\bar{z} + (x \oplus yz)$

$\bar{y}\bar{z} + (\overline{x \oplus yz})$

Class 17:

$x + y$

$x + \bar{y}$

$\bar{x} + y$

$\bar{x} + \bar{y}$

$y + z$

$y + \bar{z}$

$\bar{y} + z$

$\bar{y} + \bar{z}$

$x + z$

$x + \bar{z}$

$\bar{x} + z$

$\bar{x} + \bar{z}$

Class 18:

$x + (y \oplus z)$	$\bar{y} + (x \oplus z)$
$x + (\overline{y \oplus z})$	$\bar{y} + (\overline{x \oplus z})$
$\bar{x} + (y \oplus z)$	$z + (x \oplus y)$
$\bar{x} + (\overline{y \oplus z})$	$z + (\overline{x \oplus y})$
$y + (x \oplus z)$	$\bar{z} + (x \oplus y)$
$y + (\overline{x \oplus z})$	$\bar{z} + (\overline{x \oplus y})$

Class 19:	$(x\oplus y)+(x\oplus z)$
	$(x\oplus y)+(\overline{x\oplus z})$
	$(\overline{x\oplus y})+(x\oplus z)$
	$(\overline{x\oplus y})+(\overline{x\oplus z})$
Class 20:	$x+y+z$
	$x+y+\bar{z}$
	$x+\bar{y}+z$
	$x+\bar{y}+\bar{z}$
	$\bar{x}+y+z$
	$\bar{x}+y+\bar{z}$
	$\bar{x}+\bar{y}+z$
	$\bar{x}+\bar{y}+\bar{z}$
Class 21:	1